AUSTRALIAN Personal Computer

NEW COMMANDS FOR THE TRS-80

AUSTRALIA'S LEADING MICRO MAGAZINE

ISSUE No. 7 $1.95

COMPUTER CHESS

A roundup of recent tournaments

CONTENTS

Volume 1 No. 7 December 1980
Registered for posting as a Publication—Category B

3 NEWSPRINT
Miriam Cosic brings you the latest news in her unique style.

7 YANKEE DOODLES
A state-of-the-industry update from Tom Williams in California.

9 CHECKOUT
Super 80 . . . innovative hardware and all Australian.

11 ROBOTICS
Les Solomon with a 'think piece' on computer-controlled robots.

13 COMMUNICATION
Our readers in their own write.

17 MICRO CHESS
Kevin O'Connell reports on three recent tournaments.

21 COMPUTER GAMES
This month chess-master David Levy considers ways of guessing the odds.

27 APC-80
New commands for your TRS-80.

31 THE COMPLETE PASCAL
This month . . . records and files.

40 ZX80 WINNER
Includes the winning entries from our first Sinclair ZX80 competition.

45 LEISURE LINES
An 'Irish' crossword and a multi-coloured cube . . . both under the influence of J.J. Clessa.

45 BLUNDERS
We confess all!

46 CALCULATOR CORNER
Speed up your TRS-80 pocket computer programs.

48 FACE TO FACE
David Hebditch continues his man/machine series with free format dialogues.

51 PROGRAMS
More listings to keep you busy.

52 CALIFORNIAN JOLLY
APC's luxury tour to the West Coast Computer Faire.

59 USERS GROUP INDEX
A full listing.

Gateways to Logic will return next month.

Australian Editor
Sean Howard

Technical Editor
Ian Davies

Editor
David Tebbutt

Deputy Editor
Peter Rodwell

Advertisement Manager
Marie Pirotta

Production Manager
Michael Thomas

Typesetting
Graphic Heart
Jane Hamnell

Printed by
Mercedes Printing Co.

London Editorial Office:
Sportscene Publishers (PCW) Ltd., 14 Rathbone Place, London, W1P 1DE, Tel: 01-637 7991/2/3 Telex: 8954139 A/B 'Bunch' G. London

London Advertisement Manager
Stephen England
(01-631 1786)

Consultants
John Coll, Mike Dennis, Miriam Cosic, Michael James, David Hebditch, Sheridan Williams, Dr Adrian Stokes, Dr Stephen Castell.

Published by *Australian Microcomputer Journal*
P.O. Box 115, Carlton, Victoria, 3053.
Telephone: (03) 82-5783.
Telex: AA30333.
P.O. Box 250, North Sydney, N.S.W. 2060.

Produced in Australia
under licence from *Felden Productions Ltd.*, England.

Subscription Rates:
Australia $24.00 for 12 issues, New Zealand $30.00 for 12 issues (surface mail). Prices include postage and packing. Supplies to specialist shops can be arranged by negotiation direct with the publishers.

Cover illustration
William Stok

Distributed by
Gordon & Gotch (A'asia) Ltd., MELBOURNE, SYDNEY.

APC welcomes all unsolicited material (written, photographic & illustrative) and, although no guarantee can be given as to its safe return, reasonable care and attention will be exercised.

Guidelines for Contributors
APC welcomes articles of interest. Don't be put off if your style of writing is "under developed" . . . true worth lies in the content, and shaping features comes naturally to us! Manuscripts should not exceed 3,000 words and authors are asked to use triple spaced lines with a wide left-hand margin; diagrams, listings and/or photographs should be included wherever possible. Please enclose a stamped, self-addressed envelope if you would like your article returned.

Because of the foregoing, it is necessary to add that the views expressed in articles we publish are not necessarily those of *Australian Personal Computer*. Overall, however, the magazine will try to represent a balanced, though independent viewpoint. Finally, before submitting an article, please check it through thoroughly for legibility and accuracy.

archives

the price breakthrough in performance business computers.

Success in today's business world demands efficiency and financial accuracy. But the ever-increasing cost and comlexities of doing business are forcing you to find new ways to cut labour cost as well as gain tighter control over your business.

The ARCHIVES BUSINESS COMPUTER can give you the control you need to be successful, all in one economical desk-top cabinet.

The ARCHIVES BUSINESS COMPUTER can do your daily business functions such as accounts receivables and payables, general ledger and inventory control. It can keep track of stock on hand, stock on order, and supplies to be ordered. As a Word Processor the ARCHIVES BUSINESS COMPUTER can do your correspondence, text editing of manuals, contracts and proposals. The ARCHIVES BUSINESS COMPUTER can do virtually any information handling or record keeping operation you are presently doing plus many desirable operations that cannot economically be performed by manual methods.

The Archives disk storage can reduce entire filing cabinets to convenient 5¼ inch disks and instantly retrieve any information that you may need when you need it.

Sound incredible? The ARCHIVES BUSINESS COMPUTER is that and more!

It's a business assistant. One that never forgets, never loses reports, makes efficient use of its time and is completely trustworthy.

The ARCHIVES BUSINESS COMPUTER is also easy to work with. So easy in fact, it's no more difficult to operate than any other office machine. While the ARCHIVES BUSINESS COMPUTER is handling the extra work load, you can have more time to develop new business in the field.

There is no question that the major area restricting your business growth is in the office. This is one place where employee efficiency is still in the dark ages -requiring the handling of paper, forms and files. What you really need is not the physical presence of obsolete forms and letters, but the information and data they contain. You need the Archives Business Computer.

The Archives Business Computer offers you an economical way to individualised computing power. You can take it anywhere there's work to be done. Plug it in, turn it on, and it's working for you giving you instant access to the information you need.

Designed especially for a small business or single department in a large corporation, the Archives Business Computer is a system you can understand and use with a high degree of efficiency without being a computer specialist.

Incorporating quality and reliability the Archives Business Computer provides you with today's and tomorrow's best value in a computer system. The Archives Business Computer is a system to help your business grow.

The Archives Business Computer is backed by competent and qualified technical support for both equipment maintenance and software aid as well as continuing personalised service.

One of the best investments you can make in this inflationary economy is the Archives Business Computer. In fact, virtually every business enterprise can benefit in some practical way from business computing.

A full scale demonstration of the Archives Business Computer will help you draw positive conclusions about how your business can improve its cash flow, reduce costs and risks, improve customer service, increase sales, and improve your employee productivity.

It's time to make a move for the future.
It's time to contact Computerland
Melbourne your Archives dealer.

FEATURES
Z80 4MHz Processor
64 K RAM Standard
744K Bytes Storage (Standard)
1½ Megabytes Storage (Optional)
12" Green Phosphor Monitor
25 Lines x 80 Display
CP/M Operating System
Selectric Style Keyboard
S100 Expansion Inbuilt
Microprocessor Keyboard
240 x 100 Graphics Format
Numeric Keypad and Function Keys.

SOFTWARE
Microsoft Fortran
Microsoft Basic
Microsoft Cobol
General Ledger
Stock Control
Mail List
CBasic 2
Accounts Payable
Accounts Receivable
Property Management
Word Processing System
Microsoft Basic Compiler

ComputerLand®
in Melbourne

archives

555 Collins St. •Melbourne, Vic. •Australia 3000 Phone 62 5581 •62 6737 •Telex AA37007

Miriam Cosic and associates present the latest world-wide micro news.

MilSpec from Zilog

If you're building your own cruise missile in your garage but you're stuck for a suitable CPU with which to control it, then cheer up — Captain Zilog has come to the rescue with military specification versions of the Z80 and its supporting chippery. Now you can develop that intelligent guidance software in the comfort of your living room, using your TRS-80, knowing that it will transfer directly to the missile's CPU. And for you really advanced constructors, there are also war-going versions of the Z8001 and Z8002 16-bit processors.

If you haven't a suitable software development system, then Zilog can help with that too, for it has just knocked between 10 and 15% off its MCZ-1 microcomputers; end-user prices now start at under $6000 for the floppy disk system and at under $18,000 for hard disk systems.

Daneva controlled

Some people might have thought Daneva Control's Duoprint II printer just a little skinny with only twenty columns, even justified by the skinny price of $300.

Now they are about to release the Duoprint III, with a more useful 40 columns, for just a little bit more. Rumoured to sell for around $350, it will need a power supply, cables and a casing to live in, as extras.

One problem for many readers is that, as it stands, it's not actually compatible with the TRS-80 or some other popular computers. So to help Ian Davies, our new Technical Editor, to find his feet and become, like the rest of us, totally dedeicated to the service of our public, we're twisting his arm to come up with a method for plug compatibility.

So for around $400, a copy of APC and a bit of educated fiddling, you'll have a plain paper, 40 column, dot matrix printer. Daneva Control is on (03) 598-9207.

Handy handbook

If you're interested in microcomputers, the Australian Microcomputer Handbook is a very useful book to have on your shelves.

Its 245 pages are divided into two parts. The first goes through the fundamentals of computer design, peripherals, languages and software. There are guidelines for selection of commercial application machines, a section on word processors and one on communications and networking, plus the inevitable "future trends" chat. The second section is an alphabetical listing of manufacturers, hardware and software, pricing and local representatives — all in considerable detail.

Hopefully you won't have to wait for yours as long as I did for mine (just before the editor blithely walked in with a review copy); but they say they've got their act together. Computer Reference Guide, Suite 204, 284 Victoria Ave., Chatswood, NSW, 2067; Tel: (02) 411-2576.

Sensationally late

Speaking of getting acts together, Phd finally has their promised range of software from Sensational Software in the US. Write to P.O. Box 1007, Brunswick, Vic.

Three in one

Dick Smith has a new cassette-based software product from Microsoft for the System 80 and the TRS-80. The Editor/Assembler-Plus package, according to the blurb, does everything but water your potplants and is cheap as well.

It can assemble programs directly into memory, supports conditional assembly, recognises a whole lot of extra operators, has an automatic program origin facility, can support macros and can 'quash' the assembler and debug programs so the extra space can be used for the edit buffer. "Z-Bug", (you guessed it — the debug program) can be loaded by itself, supports up to eight break points, can use symbolic references, and has a direct calculator mode of operating.

Could one ask for more for around $40 from Dick Smith Electronics shops everywhere?

¿Qué?

You may think these translation calculators are a joke. You wait until some foreigner tries to pronounce a simple phrase such as "Do you have the right time?" from the anti-phonetic spelling of the English language, as provided by his pocket phrase book, and you will realise why the only firm with a chance in this business is Texas Instruments.

The Texas translator talks. It may sound like a machine talking, and it may be monotonous but when it speaks French, it sounds like a monotonous machine *with a French accent*, a very important point.

Panasonic has introduced a translator which doesn't talk: "Designed for use by the travelling businessman, tourist, or student," says the announcement sent to me recently, "it is ideal for vocabulary practice..." I bet. "Hello = bonjour," it says. It even detects spelling errors.

The reason for mentioning it is that this one can also be used as a calculator for simple arithmetic, plus a converter from imperial miles, feet, pounds and pints to metric kilometres and so on. So, even if the translator turns out to be useless, you haven't wasted your money.

Guy Kewney

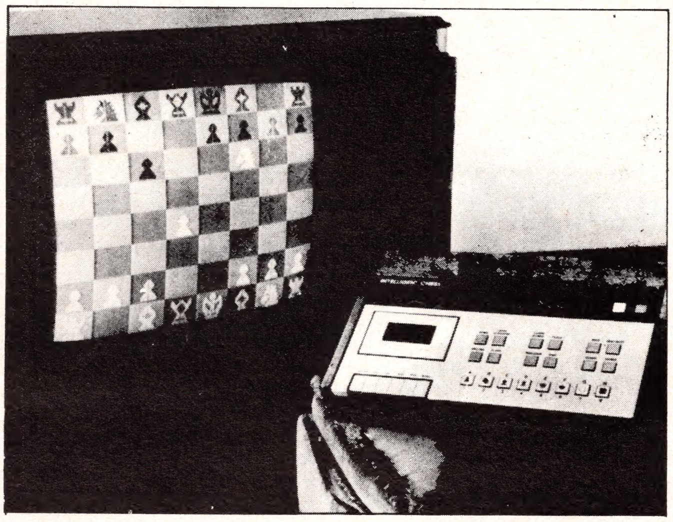
The machine that plays with itself

Chessmaster David Levy has produced a chess playing micro. He claims it's unique in eight ways.

First, it's a sophisticated game, not just a plug-in program for a video ping-pong set; it displays the chess board and pieces in full colour on a TV.

Second, it includes a built-in video cassette recorder (not one you can record Cop Shop on) to record up to 1000 full games on ordinary audio cassette tape. The player can add his/her own commentary: "This is where the machine made a fatal error," or "That error the machine made earlier now turns out to have been less fatal than I originally thought."

Third, each game comes with a free audio cassette to teach you how to operate the machine. There are other cassettes, including all the tournaments and match games ever played by Bobby Fischer, all Karpov's games (about 800 each) and all named opening variations,



Intelligent Chess, the new chess computer from Optim Games. See "The Machine That Plays With Itself".

about 300.

Fourth, any square (or squares) on the board can flash. Which may be useful in teaching chess, I suppose.

Fifth, the computer can be tortured into revealing what its next best move would have been, and compelled to make that move. And the next best, and the next, right down to the only move left.

Sixth, the machine can be stepped through a pre-recorded game, move by move.

Seventh, it can unplay a game to the point where you want to play something cleverer.

And finally, it can be left alone to play games with itself, in a shop window or at an exhibition.

It only costs 295 pounds in England, but I don't know if anyone's thinking of releasing it here. Chess enthusiasts can put their phone bills into check by calling Optim Games in England on (0279) 54547.

Visicalc etc.

Visicalc is available for the 32K Commodore micro with disk drive, for around $400, from B.S. Microcomp, 561 Bourke St., Melbourne. They've also announced a Pascal package for the same computer, at the same price, which is a powerful disk based compiler system.

Visicalc is available for the 32K Tandy from Melbourne House, at 24 Peel St., Collingwood, Vic., and they consider it very exciting news. One disk costs $195. Also exciting at the same place is the release of Galaxy Invasion and Asteroids Nova for the 16K Tandy, with sound, for $19.50.

Visicalc is available for the Apple II from Melbourne House too.

Visicalc is not available for the Compucolor, at the Logic Shop. But a word processor for the Compucolor, at $95, is.

Visicalc, by the way, is an accounts package which does sales projections, cash flow, income tax, costing estimates, wages and engineering changes.

Local liner

There's a new all-Australian high resolution graphics unit, the Super 80, available from Deforest Software for $439. It's plug compatible to the Level II Model 1 TRS-80, (CPU or interface), is completely buffered and contains its own power supply.

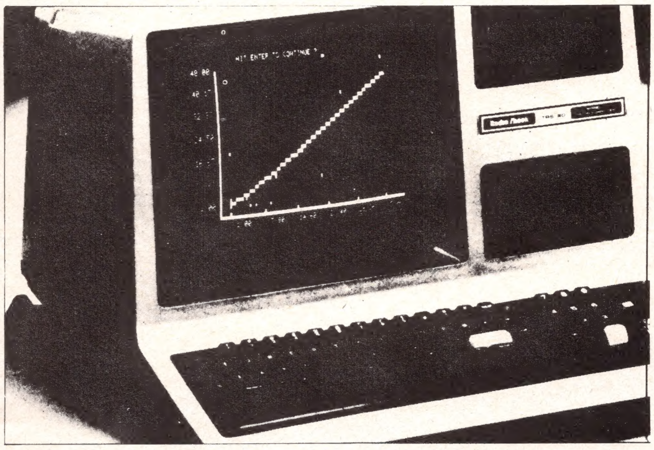
If it sells well here, there's talk of a big sales launch in the US. APC has checked it out elsewhere in this issue.

And on the subject of Deforest, they have also been appointed sole Australian distributor for the complete range of Instant Software products.

Deforest Software is at 26 Station St., Nunawading, Vic.3131. Tel:(03)877-6946.

Cromemco carrier

Computerland has added the Cromemco range of products to its stock. It will concentrate on supporting the System Three Computer, which has multi-user capabilities and comes with 64K RAM and 2.4 megabytes of disk storage as standard. As well as a range of applications and systems software for the Cromemco, Computerland will be offering training sessions for its word processing and business systems users. Further details from Computerland Melbourne, 555 Collins St., Melb. Tel: (03)62-5581.



The TRS-80 Model III is basically the Model II in a single housing. However, the disks it uses are double density and the previously optional RS232 communications interface is built in. Extra features of the Model III include a real time clock, expanded special character set, and 500 or 1500 baud cassette operation. We can expect it here about May. And incidentally, the Apple III should be here in July. Apparently, Apple is more interested in supporting the US market for the moment, and we'll see the product here when that area cools down a bit.

I thought you might like to know some more about the unavailable MZ80K mentioned last month. It has 20K of RAM (only expandable to 48K), of which the Basic takes up 14K. A 4K Monitor System in ROM can be loaded from cassette. The VDU is a 10 inch black and white, with 40 characters by 25 lines. Niceties of the system are a fairly fast Basic, built in loud speaker, and extensive graphics. One of the not so niceties is the use of cheap calculator keys of the type Commodore was quick to do away with.

Software comparisons

Evaluating software packages is no easy task — it requires plenty of staff or lots of time, and there's always the chance that, unless you produce your report very quickly, the package will have been altered in some way or have become obsolete altogether. But the Small Systems Group, headed by Larry Press and based in Santa Monica, has embarked on this onerous task and has just announced its first report, an evaluation of word processing systems. The report looks at Auto Scribe, Electric Pencil, Magic Wand and Wordstar, and costs US$12 from SSG at Box 5429, Santa Monica, CA 90405.

Guy Kewney

TCC up north

The Computer City Pty Ltd, operating under a franchise from Computer Country of Melbourne, will be opening in December at 600 Old Cleveland Rd., Camp Hill, Qld.4125. Tel(07)398-6433.

It will carry Apple, North Star, Commodore, Texas Instruments and NEC: and will offer full service and sales back up.

TCC is currently negotiating a Perth franchise, and is still looking around in Adelaide and Sydney.

Chiefly chat

When I was at the Eighth World Computer Exhibition in Melbourne. I noticed that the microcomputer stands were the second most frequented area there. The most popular place was the bar, but that's understandable. Micros are obviously taking off in Australia and those few people (mainly connected with the big machines) who aren't convinced must have different information to Informative Systems who now consider it worthwhile to have a radio advertising campaign.

Talking of advertising, did *you* realise that Peter Hatcher and City Personal Computers are one and the same?

P.S.

Have we all sent in our reader surveys? (ANS: "Yes, ma'am" or "On it's way, Ma'am")
Have we all expressed a lot of interest in "Newsprint"? (ANS: "Of course, ma'am")

1. Outlasts every competitor—200,000,000 character head warranty
2. No duty cycle limitations—even in demanding business applications
3. Professional print quality—9 x 7 matrix
4. Rugged business use construction—metal chassis—two motors
5. 80 characters per second
6. Upper and lower case—full 96 character ASCII set
7. Double width characters
8. Connects directly to TRS-80,™ APPLE® and other computers
9. Block graphics—64 shapes for charts, graphs, diagrams
10. Friction and pin feed
11. Plain paper—up to 3 parts
12. 6 and 8 lines per inch—program controlled paper savings
13. 80 and 132 columns—program controlled
14. Price—the best value in the industry. Call or write today for **information**

14 REASONS WHY TRS-80™ OWNERS CHOOSE THE MICROLINE 80

All fourteen are standard with every Microline 80. The only options are snap-on tractors and a buffered (up to 2000 characters) RS232 interface.

THE LOGIC SHOP PTY. LTD.

212 HIGH STREET, WINDSOR, VICTORIA, 3181 TELEPHONE: (03) 51 1950
91 REGENT STREET, CHIPPENDALE, N.S.W., 2008 TELEPHONE: (02) 699 4910, 699 4919

TRS-80 is a trademark of Tandy Corporation.

YANKEE DOODLES

While Australia teeters on the brink of recession, business is booming in Silicon Gulch, as Tom Williams reports.

The boom rolls on. Apple Computer has gotten big enough to cause something of a furore on that citadel of American capitalism, Wall Street. After Apple announced that it was about to launch its first public sale of stock, it rapidly became one of the hottest offerings ever. The company has been forced to ration shares to various brokers who, in turn, are giving only their best accounts first crack at buying Apple stock.

Apple is truly a business success story and is predicted to double its earnings over last year's to a whopping $300 million — up from the paltry $150 million of a year ago. What all this appears to mean for the personal computer market is that we will be seeing even more Apples and Apple-related products than the veritable flood to which we have become accustomed. The sale of stock will, for one thing, provide good revenue for continued expansion and the attention Apple has gained in the business world may well put it in a position to challenge Tandy for the title of top dog. There has even been a series of radio commercials by Dick Cavett *(not the Dick Cavett, surely? — Ed)* promoting Apple computers.

All, however, is not well in Apple Land. The long-awaited Apple III, which was debuted last May at NCC, has still not shown up on dealers' shelves. There were rumours about soldering problems in the power supply that forced the company to delay volume production for a while.

"All... is not well in Apple Land."

On the forefront of new technology we once more find Zilog. After having overcome what appears to be the last hurdles for the Z-8001 (the Z-8000 version with segmented addressing that accesses eight megabytes), Zilog is reported to be working on a Z-9000. The Z-9000 will not have a new 32-bit architecture but will combine the Z-8000 16-bit architecture with virtual memory support on a single chip.

Virtual memory is a technique whereby the CPU creates 'virtual addresses' for all data, which it treats as if they were all a part of one continuous memory. It then sends these virtual addresses to a memory management unit (MMU) which converts them into physical addresses, which may exist in RAM or on disk. National Semiconductor had earlier presented its 16000 family of 16-bit CPUs, one member of which combined with an MMU chip to support virtual memory. If Zilog does come out with the Z-9000, it will be the first to combine CPU and MMU on a single chip.

The reason for all this activity in the area of greater and greater memory is the demand of the market for multiuser systems and distributed computing. Manufacturers are pursuing all approaches, but three watchwords are: modularity, expandability, and compatibility. These are sometimes, but not always, satisfied by massive amounts of RAM under a single CPU. At the low end (one to four users) which is admirably served by such manufacturers as Altos, a moderate amount of RAM supports the users and the systems offer expandability in the direction of disk storage.

A new entry has just joined this area in the shape of Morrow Designs' Decision 1 System. Morrow has combined a long experience in S100 systems with a range of Winchester disk technology to produce a line of computers that are almost continuously expandable from single-user floppy based configurations with between 200k and 2 Mbytes of disk storage up to a multi-user system with 256k of RAM and 28Mb of hard disk, priced at just over $14,000.

The Z80-based Morrow system offers memory management hardware (not to be confused with virtual memory above) that has a memory map to support up to 16 tasks with complete memory protection and automatic memory allocation. In addition, a UNIX-like operating system, uNIX, and CP/M are offered such that the new uNIX can communicate with CP/M.

Another new entry which showed up at the recent Wescon show held in Anaheim, California in September is a rather novel machine by Piiceon Corp. of San Jose. The Sword computer (not to be confused with the long defunct Japanese Sord) is one of the first stand-alone systems to use the Intel 8086. The most striking thing about the Sword-100 is its CRT display, which is a 15-inch tube mounted 'on its side'. With this arrangement, the machine can display up to 66 lines of 80 characters each, a particularly nice feature for word processing. Memory consists of up to 8 kbytes of ROM and 64 kbytes of RAM. Disk storage consists of two Qume double-sided, double-density drives giving a total of 2 Mbytes.

But — and here is a point once again indicative of the trend in the industry — it is designed to be an intelligent terminal workstation which can also fit into a distributed processing local network.

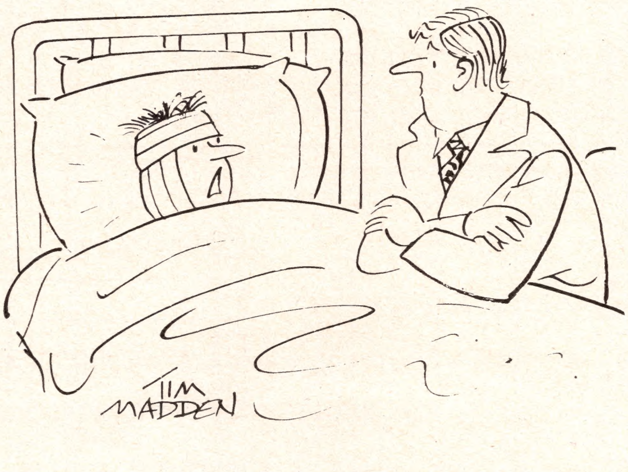
These local networks — individual processors sharing more expensive resources such as disks and printers — are starting to show up everywhere. The Bank of America has recently purchased a Nestar Cluster/One with *56* Apples for use as a preprocessor for entering cheques. The results are of course then fed into the bank's IBM behemoth for further digestion. According to Nestar's president, Harry Saal, the long touted 'office of the future' is not just around the corner — it's already here with a vengeance, and almost without anyone noticing.

"...who's using those 64k RAM chips..."

Everyone is asking "who's using those 64k RAM chips in a real product?" The answer is Hewlett-Packard. To be sure, the HP-1000 L Series is not exactly your garden variety personal computer but it should not take long for the technology to spread to personal computers, especially when the results of some of HP's tests become better known, and prices come down with volume sales.

HP recently introduced a 312k RAM board using the new 64 kbit chips. During reliability tests, technicians found the 64k chips to be more reliable than conventional 16 kbit memory chips. This obviated the need to incorporate costly error-correcting circuitry into the new memory system. Until recently, one of the most important problems with memories of this density was soft errors introduced when random alpha particles strike memory cells. Newer techniques in IC fabrication and technology are reducing these errors by a factor of ten over their previous rate. As a result memories should continue to become denser, and faster with a per-bit cost rollback of 20% to 30% per year. *(I think he means they'll get cheaper — Ed.)*

The boom rolls on!



"And she threw everything at me — including the computer aimed at the home user!"

COMPUTER REFERENCE GUIDE

Suite 204, 284 Victoria Ave.,
Chatswood, N.S.W. 2067
AUSTRALIA.
Tel.: (02) 411 2576

AUSTRALIAN MICROCOMPUTER HANDBOOK

The MICROCOMPUTER HANDBOOK is a unique Australian Publication that reviews the large range of microcomputers currently being sold in this country.

PART I includes a wealth of useful information on microcomputer theory. Chapters 1, 2 and 3 introduce microcomputers, outline the differences between microprocessors, microcomputers and microcomputer systems, and discuss peripherals and software capabilities. Chapter 4 describes some of the popular microcomputer Software Packages including the CP/M and MP/M operating systems from Digital Research, and various high-level languages and utilities from Microsoft, Micro Focus, MicroPro, Structured Systems Group and others. Various popular application packages including word processing are also described.

Chapter 5 outlines the capabilities found in BASIC – the most popular of all microcomputer languages. Chapter 6 provides guidelines for selecting microcomputers, particularly for those people contemplating commercial applications. Chapter 7 looks at the capability of microcomputers in word processing and is also a useful introduction to word processing in general. Chapter 8 projects the future potential of microcomputers in the coming years, while Chapter 9 looks at the future trends in microprocessing and microcomputing design, as outlined by a leading semiconductor manufacturer.

Chapter 10 discusses the use of microcomputers in communications and networking applications and is also a useful introduction to the general area of communications and distributed data processing. Chapter 11 traces the history of the S-100 Bus and outlines the different bus signals that have nowadays become fairly standardised.

PART II provides a 2 to 6 page summary on the range of microcomputer and desktop computers sold in Australia including the following:

• AI Electronics • ADE Datasystems • Alliance Digital (ERA) • Alpha Micro Systems • Altos • Apple • Cado • Commodore • Compucolor • Compucorp • Cromemco • Data General • Datapoint • Diablo • Digital Electronics • Digital Equipment • Durango • Electronic Control Systems • Exidy • Fairlight Systems • Findex • Hewlett-Packard • IBM • Intel • Intelligent Systems Corp. • Intertec • John F Rose • J.T. Microcomputers • Logical Computer Systems • Micro V Corp. • Micromation • Mitsubishi • Motorola • National Panasonic • National Semiconductor • North Star • Ohio Scientific • Omni • Ontel • Onyx • Philips • Small Business Computer Company • Sord • System 80 • Tandy • Tektronix • Texas Instruments • Vector Graphics • Versatile 4 • Wang • Zenith • Zilog

---

I would like to order the Computer Reference Guide MICROCOMPUTER HANDBOOK.
The cost of the HANDBOOK is $22 (which includes $2 postage).

Name ____________________

Company ____________________

Address ____________________

____

Postcode ____________________

Telephone ( ) ____________________

☐ Cheque Enclosed OR Credit Card Number

[ ]

☐ Bankcard ☐ Diners Club

☐ American Express

Signature ____________________
(for credit card)

Expiration Date ____________________

Please forward to: COMPUTER REFERENCE GUIDE,
204/284 Victoria Ave, CHATSWOOD NSW 2067

AMJ. 12/80

CHECKOUT

SUPER 80

Benchtests aside for this issue, as Ian Davies takes a look at a high resolution board for the TRS-80.

Introduction

Customtronics' high resolution board, the Super 80, is a memory mapped device which generates 384 horizontal by 192 vertical dots. It plugs directly into the TRS-80 or its expansion interface to enable the high resolution plus full alphanumeric upper/lower case facilities.

As the owners of Apple and Sorcerer computers would be quick to boast: high resolution improves the effectiveness of sales graphs in business. Scientists and engineers can use displayed figures, schematics and logic diagrams far more effectively. Architects can draw detailed plans with meaningful three dimensional views. And, of course, graphics in games add a great deal of realism.

Hardware

Installation of the Super 80 doesn't require any modification to the TRS-80 and so shouldn't effect existing warranties. It simply plugs into the edge connector of the TRS-80 or its expansion interface. No power drain is imposed on the TRS-80 as Super 80 contains its own power supply plus the required protection against voltage spikes for its 47 ICs. The circuitry appears to have been carefully designed with no sign of bodges that are to be seen on some boards.

The unit is fitted with a fully buffered expansion connection which allows other devices to be chained into the Tandy edge connector. Super 80 doesn't interfere with any TRS-80 port usage and so will not interfere with any programs that require ports to have certain values. Because it does not prevent normal TRS-80 operation, it can be left permanently connected to the computer. Conversion back to normal resolution is a matter of returning the video DIN plug from the Super 80 socket to the TRS-80 keyboard.

To the processor the high resolution board looks like a 1k chunk of RAM at the top end of memory. This is used to store 64 programmable characters each occupying an area equivalent to a normal character. A second 1k of RAM in the Super 80 is an exact replica of the video RAM of the TRS-80.

The standard TRS-80 display provides 16 rows of 64 characters. In graphics mode, a character space is divided into six pixels (Fig. 1) giving a maximum graphics resolution of 128 horizontally x 48 vertically. The total number of combinations in each six cell graphics character is therefore 64 (2^6), which are designated by Tandy as CHR$(128) to CHR$(191).

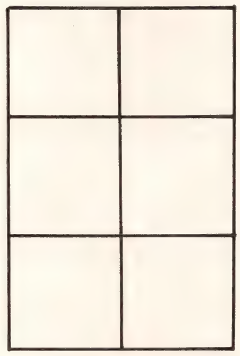


Fig. 1 Normal TRS-80 graphic block divided into six pixels.

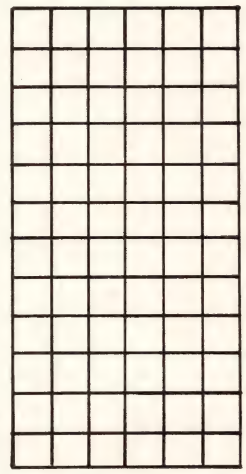


Fig. 2 Super 80's high resolution graphic block; same area as the normal block but divided into 72 pixels.

Super 80 adds 64 user-defined graphics characters which divide a normal character into 72 pixels (Fig. 2). As this allows 2^{27} possible dot combinations (approx. 4.7×10^{21}) it is impractical to allocate each a predefined reference number. Instead, Super 80 allows 64 programmable graphic characters which are given values from CHR$(192) to CHR$(255). Don't worry — these values will still perform their normal TAB functions if, for example, PRINT CHR$(122) is used.

In a standard TRS-80, POKEing CHR$ (192) to CHR$ (255) into video RAM will produce the same result as the values from 128 to 191. However, Super 80 will interpret these values according to the preprogrammed configurations in its first 1k of RAM.

Except in the case of a 48K TRS-80, all RAM locations are unaffected by Super 80. Because the 1k of RAM used to store the high resolution configuration is at memory locations 63488 to 64496, it conflicts with the top 1k of a 48k machine. A slide control switch is provided to disable Super 80's first 1k, and prevent double addressing in such cases.

As mentioned previously, Super 80 is not a port driven device. Besides the obvious advantage of not tying up valuable ports, it means that it will generally use less program memory in performing tasks associated with high resolution graphics. Once the required character patterns are defined in memory, they can be recalled as often as necessary with a simple POKE. For example, suppose a small box of character size needs to be drawn on the screen many times during the execution of a program. It need be defined only once, allocated a character number and then POKEd into video RAM as required. On the other hand, a port operated unit would have to redraw the box each time it is required, using additional RAM to store the appropriate subroutines.

Software

At the moment, there's a Basic program, HIGOS, which must be loaded from cassette or disk. It defines the high resolution characters and stores them in Super 80's 1k of RAM. Unfortunately, HIGOS cannot be used in conjunction with user programs so, while redefining high resolution characters can be done manually, it's somewhat tedious. But then, I wouldn't have expected defining one of 2^{27} combinations to be anything less!

I'm told it will not be long before a machine language version of HIGOS becomes available. It will be accessible from user Basic programs as well as from the command mode. So not only won't you have to dump HIGOS before running your own Basic program but reproducing characters during program execution won't chew up as much processor time.

The Basic HIGOS plots a magnified graphics character on the VDU. Using cursor controls, pixels can be filled or cleared as required and the character given a value of between 192 and 255. Modification of previously defined characters is simple: type in its reference number and the magnified graphic block will reappear. From there it's just

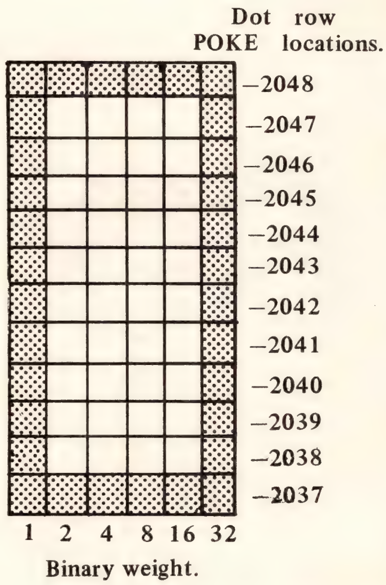


Fig. 3
Example of CHR$ (192) defined as a box (shaded area is illuminated).

as if you were defining the character for the first time. If HIGOS is cleared from memory, the characters will remain in Super 80's memory until cleared or power is switched off.

To illustrate the procedure for programming a character manually, assume that CHR$ (192) has been chosen to represent a small box. Each dot row of CHR$ (192) is given its own memory location which can be POKEd or PEEKed. Each dot column of a character is allocated a value (called a binary weight). Figure 3 gives details of the box programming.

Defining CHR$ (192) as a box requires the following commands: POKE —2048, 63, POKE —2047 to —2038 with 33 and POKE —2037, 63. Dot row 1 needed every dot position turned on and so a value of 1+2+4+8+16+32 = 63 was POKEd into that position. The sides of the box needed only the 1 and the 32 columns turned on and so POKE byte position, 33 was used.

Features against Super 80 are that it interferes with the Level II commands POINT, SET and RESET as the interpreter has some difficulty deciphering the contents of a character block in video memory with a value of 195. Also some machine language programs (e.g. Sargon II) POKE values of between 192 and 255 expecting normal TRS-80 graphics characters. Instead this results in the incorrect high resolution characters.

Conclusions

It's interesting to see the TRS-80's limitations being overcome by an ever increasing number of peripherals, add-ons and modifications. And it's nice to see one of the most practical of these coming out of Australia.

Lower case with descenders, Chinese letters and electronic and scientific symbols are now all possible on the TRS-80 with a resolution that compares favourably with Apples' 280 by 192 and Sorcerer's 512 by 240.

Against the Super 80 are the loss of some Basic statements and a very slight decrease in program execution speed. Plus features are TRS-80 plug compatibility; no need for hardware modifications; a price of $439; and that it's made in Australia.

BARE BONES OF ROBOTICS

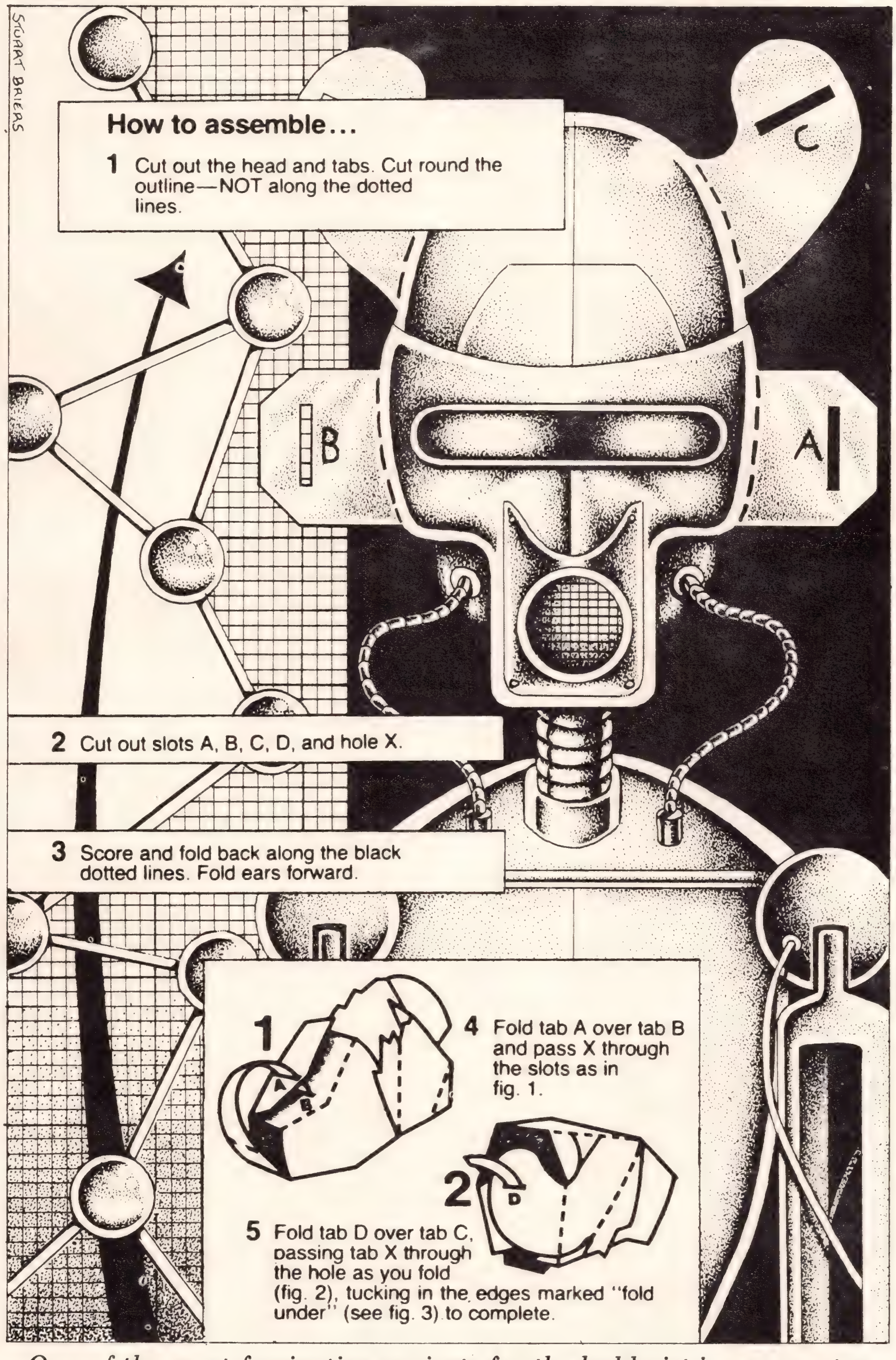


Illustration by Stuart Briers

One of the most fascinating projects for the hobbyist is a computer-controlled robot. Leslie Solomon, technical director of the US magazine Popular Electronics, *passes on some tips and ideas.*

I'll begin by telling you the story of the guy who went into the pizza parlour and ordered a large pizza. The man behind the counter asked him whether he should cut the pizza into six or eight slices. The customer said, "Cut it into six slices because I can't eat eight!"

The same is true of creating a robot — we have two large pieces, large enough so that we couldn't 'eat' them both. One half is the mechanics of robotics while the other half is computer. It's difficult enough to build complex mechanics, rather than simultaneously having to sweat out a computer and its software.

There are two steps in creating a computerized robot: first it should be mechanically constructed and tested before installing the computer, then it should be mated with the computer, after the computer and its software have been tested.

Building the physical elements of a machine is just as difficult as constructing the computer interfaces and writing the software. The question is like asking which came first — the chicken or the egg? Do we build the robot then the computer, or vice versa? Or do we build both together, and what do we do if the damn thing doesn't work? How do we decide where the problem is? I determined that we needed a separate electronic and mechanical approach and this is what we did.

Since we should be able to test the various physical elements of the robot as construction goes along, what's needed is a simple, low-cost, yet effective means to perform the tests, that also will emulate the future computer. In a real sense, we should first create our mechanical man analogue and have full control over it before we install the 'intelligence'. Let's first take a look at a simple remote control.

The most common and usually the first thing that comes to mind is radio control, using the same systems as model planes -- provided you have enough channels. However, this involves RF transmissions, the need for a licence, and a possible visit from government-type people who usually have no sense of humour. There are a couple of other ways that don't involve RF and the need for a licence, and they remove the probability of external noise or signals causing unwanted problems.

The first approach is called Induction Transmission and can be built by anyone. Unless two experimenters work in the same building, there will be no cross interference, and you can have quite a number of signalling channels going at the same time.

The technique is simple. All you have to do is wind a turn or two of conventional insulated wire around the area where the robot experiments are to take place. In my case, I wound my turns along the junction of the walls and ceiling of my workshop. It makes no difference how small or large the area is as long as the total resistance of the wire is not excessive; you can wind the turn around an acre if you so desire. This turn forms the primary of what will wind up as a strange-looking transformer.

This turn of wire is connected to the output of a conventional audio amplifier — the power requirement depends on how large the loop is, but a 5 or 10 watt amplifier will usually suffice. The turn then simulates the speaker load. The secondary of the transformer is formed by using a lot of turns a few inches in diameter, with this coil connected to a small transistorized battery powered audio amplifier.

To test the basic system, feed a conventional music source to the loop amplifier and connect an earphone or even a speaker to the portable amplifier of the secondary. As long as you keep the portable secondary within the large primary turn and orient the secondary so that it is parallel with the primary, you should hear the music. The magnetic field is strongest within the turn, but it also extends outwards a little; if the turn is on an upper floor the music will be heard one or more floors above and below the loop.

Some of you will probably realise

that this magnetic induction technique can be used to listen to a silent radio (a boon for those of you that have teenage children who play their music at deafening volumes), or can even be used as a hearing aid for those who need it, without the constraints of having any interconnecting wires.

Now, suppose you build an array of audio tone generators, each operating at a different audio frequency and whose outputs can be connected to the input of the loop power amplifier. Each generator has a pushbutton switch that turns its associated tone on and off.

At the receiver there's a companion array of tone decoders following the audio amplifier — the 567 PLL is a good choice. The output of each tone decoder operates a simple transistor switch that opens or closes a relay. A look at the 567 specs will show several circuits that can be used.

Each relay controls the supply power to a particular motor; as each originating pushbutton switch is operated, the associated remote relay operates. Latching flip-flops can be arranged to maintain a tone for some period if desired.

That then is a simple yet powerful remote control system which can be used anywhere without causing any electronic problems. In fact, using this technique, it's possible to build and operate a complex robot without having a computer.

Now to the robot. Since it's always best to start at the beginning, the first consideration should be total robot movement. This usually requires consideration of how to obtain forward, backward, right/left rotation, sideways motion (if desired), and any other direction you see fit. Then you have to consider the degrees of freedom for each limb — these are the joints. You can have hinge, rotation and telescope and combustions of the above.

Two very important items we discovered at great cost — be sure to install limit stops on all mechanical motion and don't make the arms too long! It's amazing how much damage can come when cables and PC boards get ripped out by excessive rotational zeal, and how easily things get bent out of shape by heavy robots falling on them.

You, the builder, have to consider whether to use driven wheels, tracks or make articulated legs. Keep in mind that wheels and slender narrow tracks are are good only in a billiard table type of environment — rough or soft surfaces may cause the machine to hang up, water can cause shorts, and mud is death.

After deciding on the means of motive power, it's best to start with the baseplate that mounts the drive motors, their battery, and the remote control receiver with one (or more) channel relays controlling motor power. Add sufficient weight to the baseplate to simulate the estimated final weight of the robot.

Using the master pushbutton switches on the tone system, you can now test the mechanism. Drive the baseplate around the area and thoroughly check its operation as to turns, forward and backward operation, and stopping. The latter is very important!

By using the loop technique, you can now construct any types of limbs or other mechanical items and test them merely by connecting them to the receiver tone decoders. The testing does not have to take place on the robot itself. The cost of the small receivers is so low (just a loop, an OP amp and a tone decoder) that you can build a few of them, each with only one or two tone decoders, just for the element you are testing. At one time we were playing with three different items at the same time while our robot baseplate was 'taking a walk' around our workshop using a computer program which in turn operated our computer to turn on the various tones.

It'll be useful at this point to make another small diversion.

If you consider a robot having a planned weight of 100 lbs and designed to travel at about 20 mph, or to carry a 300 lb load at 5 mph, you'll need about 5½ HP, or some 5kW of power, or over 300 amps from a conventional 12 volt battery. If you use four 17 lb car batteries, and add the weight of the motors, mechanics, etc, you will have reached 100 lbs. If you cut the top speed to 10 mph, and the top weight to 200 lbs (100 machine + 100 load), you can cut down to two batteries. So you had better sharpen up your pencil, break out the old physics book, and do a lot of paper work before construction.

Give some thoughts to the installation of a small solid-state TV camera so you can 'see' what the robot is up to.

Now for the computer. Essentially, all the computer has to do is turn on (and Off) a simple transistor relay driver that now substitutes for the loop-driven relays. Since most computers have 8 data bits, you now have the means to turn on 256 different relays. It's not difficult to create a logic tree, so arranged as to turn on any selected relay with one particular set of bits; 256 discrete functions are probably enough to run even a complex robot.

Software can either be written for the computer, or you can start with something like Dr Li Chen Wang's 'Robot' language as published in volume 2, issue 8 (number 18) of *Dr Dobbs Journal*, or John Webster's *Robot Simulation On Microcomputers* that appeared in the April 1978 issue of *Byte* magazine.

We used Li Chen's program with a little modification because like the computer-emulating tone loop just described, you don't need a robot to use this software. The cursor on the video display substitutes for the actual robot and you can use the software to guide the cursor through its paces. The software enables the cursor to walk around obstacles. In real life, the signals driving the video cursor can be used to drive the robot.

So, now you are in an interesting position since it's possible to build and operate a robot without a computer, and you can design and test software without a robot. After all is cleaned up, you can combine the two.

Then there is the question of what the computer should do in the robot. Unfortunately, it's been found that when you consider all the things that you would like the robot to do, you run out of computer — unless you install a disk system. With one thing leading to another, you suddenly realize that your computer will start to look like a tank and require submarine batteries.

There's *no* right answer as to what the robot computer should do; different people have different ideas. My answer was simple: why not have the robot computer do only internal 'housekeeping' — watching over battery level, robot mechanical component positioning, contact sensing, and other internal tasks. But what about the rest?

I thought that since my main computer had 56k of RAM, a dual 8 inch disk system, a light pen, an excellent video display and loads of software (in which I had Li Chen's language running), why not use it to control the robot by treating it (the robot) as a high-speed (19 kbaud) serial port? Using the light pen, we could sketch the area in which we wanted the robot to do things, include obstacles such as chairs, tables, walls, etc, and have the cursor wander along its course, avoiding the obstacles and behaving as if it had sense.

We even considered installing a small CB transceiver in the robot, so it could communicate with anyone it met during its travels. Of course, the computer operator, having the other transceiver, would perform this miracle.

We found that although the tone link worked fine, what we needed was a really high-speed link that wasn't bandwidth limited like the audio system. It was about this time that I saw a German audio system —built by Senheiser — that uses a bank of infra-red LEDs to talk to an infra-red detector mounted in a set of headphones. The audio reception was great and I decided to try this technique.

To experiment with this optical data link we built a pair of high-speed ultraviolet transmitters using a few UV LEDs in parallel, arranged so that their optical polar diagrams overlapped to produce a broad fan-like beam that would cover a wide area. One was used as the computer receiver and the other as the robot receiver and a conventional serial interface was installed on the robot computer. Thus, our robot now be passed back and forth between the host computer and the computer resident in the robot. At 19 kbaud we found that lots of data could be passed back and forth.

Of course, using this technique somewhat limited the range the robot could traverse even though we found out that the UV link could go around reasonable corners. The built-in TV camera, having its own RF link back to our TV receiver, enabled us to see what the robot was up to in its wanderings.

This approach may violate your concepts of what a robot is and how it should be controlled. But the robot, the induction loop and the optical system worked fine and we had a lot of fun. We've learned a lot — both from a mechanical and electronic viewpoint — and I guess that's what this computer stuff is all about.

More, please

Let me first congratulate you for your fine magazine; I confirm this opinion by enclosing my subscription form.

I have one gripe though. I own an Apple and appreciate the programs published so far for this machine, but would enjoy more. I realise the difficulty in presenting a diverse range of software especially in relation to (I suppose), your committment to the more popular machines such as the TRS-80, but feel a specialist page for at least the TRS-80, Apple and PET might take the burden off the "Programs" Section.

B. Phillips, Vic.

We've beaten you to it! Well, at least a start has been made with the 'APC-80' for Tandy users. And the growing number of Sinclair ZX80 owners can look forward to a regular page of their own beginning next month.

We're currently thinking about a similar page for Apple users while a general TRS-80/System 80 column is firmly on the drawing boards (contributions to Ian Davies, Technical Editor, please).

Naturally such specialist columns rely a good deal on reader contributions so we'll be on the lookout for interested parties to participate.

All this doesn't mean 'Programs' will bite the dust. When the big boys move over, it'll be free to present a much broader range of software than before. So keep your contributions coming and we'll point them to the appropriate column – Ed.

16-bit efficiency

Guy Kewney repeats a red herring about 16-bit processors. They do *not* need twice the memory size of that used by the eight-bit kind. In fact averaged over real programs they are likely to use less memory to complete a given task as each instruction is more efficient. The Z8000 uses only two bytes for those instructions likely to be used most. Typical eight-bit programs use around 1.8 bytes per instruction and more of them.

The S100 bus uses a memory technique that permits both 8 and 16 processors to use the same memory in a multiple processor system. The method could easily include 32 bit processors and so permit the same system to run programs written for all popular processors. The E78 bus has a similar facility.

R. Silson, N.S.W.

Cut it out

My employer has recently taken delivery of an Apple II with two floppy disk subsystems. Since the manuals state that only one side of the disks can be used I was surprised to find that a disk containing demonstration programs for the Supertalkers peripheral had data recorded on both sides. There were no problems in reaching this data.

On trying to write data to the "back" of an Apple disk I got a message indicating that the disk was write-protected. This was because the write-protected cutout is on the wrong side and an examination of the Supertalker disk showed that it had cutouts on both edges of the disk casing. I carefully cut another write permit cutout opposite the one already on a BASF floppy and found that I could use the second side without any trouble.

It seems ridiculous to me that one side of a floppy should be rendered useless simply through the lack of a hole punched in the disk case.

D. Watson, N.S.W.

We definitely do not advise readers to follow Mr. Watson's example. Manufacturers test both sides of each disk; if one side is faulty then it's sold as a single-sided disk, using only the good side; so any data recorded on the reverse side of a single-sided disk could be corrupted. Now it's possible that, to satisfy the demand for single-sided disks, some are sold which are fault-free on both sides; however you can't take advantage of this because inside the disk's protective envelope there's a wiper which collects any dust or dirt from the disk's surface as it revolves; inserting the disk the other way round to use the reverse side will, by reversing the disk's rotation, spread the collected dirt over the good side – and then you've really got problems! For the same reason, you shouldn't try using a proper double-sided disk in a drive designed for single-sided disks. – Ed.

Unique in concept—the home computer that grows as you do!

The Acorn Atom

Special features include
- ★ FULL SIZED KEYBOARD
- ★ ASSEMBLER AND BASIC
- ★ TOP QUALITY MOULDED CASE
- ★ HIGH RESOLUTION COLOUR GRAPHICS★

★ optional

$780

incl. Sales Tax, Packaging and Postage

INCLUDES FREE 122 PAGE BASIC MANUAL

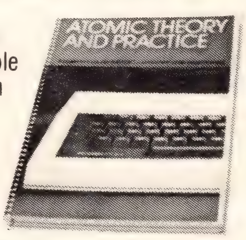
HOBBY! OFFICE! EDUCATIONAL!
- In the Home
- In the School
- In the Business

ACORN COMPUTER

The Acorn Atom is a definitive personal computer, simple to operate. A powerful, full facility computer with all the features you would expect.

Just connect the assembled computer to any domestic TV and power source and you are ready to begin. (Power requirement: 8V at 800mA).

There is an ATOM power unit available — see the coupon below.



FREE MANUAL

Free with every ATOM, is a computer manual. The first section explains and teaches you BASIC, the language that most personal computers and the ATOM operate in. The instructions are simple and learning quickly becomes a pleasure. You'll soon be writing you own programs. The second section is a reference manual giving a full description of the ATOM's facilities and how to use them. Both sections are fully illustrated with example programs.

The standard ATOM includes:

HARDWARE

★ Full-sized QWERTY keyboard ★ 6502 Microprocessor ★ Rugged injection-moulded case ★ 4K RAM ★ 8K HYPER-ROM ★Audio cassette interface★VHF TV output★

SOFTWARE

★ 32-bit arithmetic (±2,000,000,000) ★ High speed execution ★ 43 standard/extended BASIC commands ★ Variable length strings (up to 256 characters) ★ String manipulation functions ★ 27 × 32 bit integer variables ★ 27 additional arrays ★ Random number function ★ PUT and GET byte ★ WAIT command for timing ★ ★ DO-UNTIL construction ★ Logical operators (AND, OR, EX-OR) ★ Link to machine — code routines ★ PLOT commands, DRAW and MOVE ★

The ATOM modular concept

The ATOM has been designed to grow with you.

As you build confidence and knowledge you can add more components. For instance the next stage might be to increase the ROM and RAM on the basic ATOM from 8K + 4K to 12K + 12K respectively.

The 12K + 12K ATOM will give you a direct printer drive, floating point mathematics, scientific and trigonometric functions, high resolution graphics.

From there you can expand indefinitely. Acorn have produced an enormous range of compatible PCB's which can be added to your original computer. For instance:

★ A module to give red, green and blue colour signals ★ Teletext VDU card (for prestel and Ceefax information) ★ An in-board connector for a communications loop interface — any number of ATOMs may be linked to each other — or to a master system with mass storage/hard copy facility ★ Floppy disk controller card. For details write to the address below.

Your ACORN ATOM may qualify as a business expense. To order complete the coupon below and post. Sinclair Equipment and Acorn Computers are a division of Consolidated Marketing (Imports) Pty. Ltd. **All components are guaranteed with full service/repair facility available.**

Please send me the following items:

Quantity	Item		Item price inc. Sales Tax	TOTALS
	ATOM ASSEMBLED—8K ROM + 4K RAM (MIN)		$780	
	ATOM ASSEMBLED—12K ROM + 12K RAM (MAX)		$1380	
	1K RAM SETS		$28	
	4K FLOATING POINT ROM (inc. in 12K Version)		$116	
	PRINTER DRIVER	6522 VIA	$46	
	(inc. in 12K version)	LS244 Buffer	$9	
	MAINS POWER SUPPLY (1.3 amps)		$30	
	EXTERNAL BUS PASS		$45	
			TOTAL	

ORDER FORM:

SINCLAIR EQUIPMENT (AUSTRALASIA) PTY. LTD.

Show room: 308 High Street, Kew 3101, Victoria. Telephone: 861 6224.

Name ____________________

Address ____________________

____________________ Postcode __________

I enclose cheque/Bankcard/Diners Club/Amex

CMC6

NEWCOMERS-START HERE

What follows is a brief guide for the microcomputing novice. It has been designed for quick reference — with all the key words in bold type; of course if you're feeling adventurous, you're welcome to read it right through. Whichever way, we trust you will find it helpful. Happy Microcomputing!

Welcome to the confusing world of the microcomputer. First of all, don't be fooled; there's nothing complicated about this business, it's just that we're surrounded by an immense amount of necessary jargon. Imagine if we had to continually say "numbering system with a radix of sixteen in which the letters A to F represent the values 10 to 15" when instead we can simply say "hex". No doubt soon many of the words and phrases we are about to explain will eventually fall into common English usage. Until that time, APC will be publishing this guide — every month.

We'll start by considering a microcomputer's functions and then examine the physical components necessary to implement these functions.

The microcomputer is capable of receiving information, processing it, storing the results or sending them somewhere else. All this information is called **data** and it comprises numbers, letters and special symbols which can be read by humans. Although the data are (yes, it's plural) accepted and output by the computer in 'human' form, inside it's a different story — they must be held in the form of an electronic code. This code is called **binary** — a system of numbering which uses only 0s and 1s. Thus in most micros each character, number or symbol is represented by eight binary digits or **bits** as they are called, ranging from 00000000 to 11111111.

To simplify communication between computers, several standard coding systems exist, the most common being **ASCII** (American Standard Code for Information Interchange). As an example of this standard, the number five is represented as 00110101 — complicated for humans, but easy for the computer! This collection of eight bits is called a **byte** and computer freaks who spend a lot of time messing around with bits and bytes use a halfway human representation called **hex**. The hex equivalent of a byte is obtained by giving each half a single character code (0—9,A—F): 0=0000, 1=0001, 2=0010, 3=0011, 4=0100, 5=0101 ...... E=1110 and F=1111. Our example of 53 is therefore 35 in hex. This makes it easier for humans to handle complicated collections of 0s and 1s. The machine detects these 0s and 1s by recognising different voltage levels.

The computer processes data by reshuffling, performing arithmetic on, or by comparing them with other data. It's the latter function that gives a computer its apparent 'intelligence' — the ability to make decisions and to act upon them. It has to be given a set of rules in order to do this and, once again, these rules are stored in memory as bytes. The rules are called **programs** and while they can be input in binary or hex (machine code programming), the usual method is to have a special program which translates English or near-English into machine code. This speeds programming considerably; the nearer the **programming language** is to English, the faster the programming time. On the other hand, program execution speed tends to be slower.

The most common microcomputer language is **Basic**. Program instructions are typed in at the keyboard, to be coded and stored in the computer's memory. To run such a program the computer uses an **interpreter** which picks up each English-type instruction, translates it into machine code and then feeds it into the **processor** for execution. It has to do this each time the same instruction has to be executed.

Two strange words you will hear in connection with Basic are **PEEK** and **POKE**. They give the programmer access to the memory of the machine. It's possible to read (PEEK) the contents of a byte in the computer and to modify a byte (POKE).

Moving on to **hardware**, this means the physical components of a computer system as opposed to **software** — the programs needed to make the system work.

At the heart of a microcomputer system is the **central processing unit (CPU)**, a single microprocessor chip with supporting devices such as **buffers**, which 'amplify' the CPU's signals for use by other components in the system. The packaged chips are either soldered directly to a **printed circuit board (PCB)** or are mounted in sockets.

In some microcomputers, the entire system is mounted on a single, large, PCB; in others a **bus** system is used, comprising a long PCB holding a number of interconnected sockets. Plugged into these are several smaller PCBs, each with a specific function — for instance, one card would hold the CPU and its support chips. The most widely-used bus system is called the **S100**.

The CPU needs memory in which to keep programs and data. Microcomputers generally have two types of memory, **RAM** (Random Access Memory) and **ROM** (Read Only Memory). The CPU can read information stored in RAM — and also put information into RAM. Two types of RAM exist — **static** and **dynamic**; all you really need know is that dynamic RAM uses less power and is less expensive than static, but it requires additional, complex, circuity to make it work. Both types of RAM lose their contents when power is switched off, whereas ROM retains its contents permanently. Not surprisingly, manufacturers often store interpreters and the like in ROM. The CPU can only read the ROM's contents and cannot alter them in any way. You can buy special ROMs called **PROMs** (Programmable ROMs) and **EPROMs** (Eraseable PROMs) which can be programmed using a special device; EPROMs can be erased using ultra-violet light.

Because RAM loses its contents when power is switched off, **cassettes** and **floppy disks** are used to save programs and data for later use. Audio-type tape recorders are often used by converting data to a series of audio tones and recording them; later the computer can listen to these same tones and re-convert them into data. Various methods are used for this, so a cassette recorded by one make of computer won't necessarily work on another make. It takes a long time to record and play back information and it's difficult to locate one specific item among a whole mass of information on a cassette; therefore, to overcome these problems, floppy disks are used on more sophisticated systems.

A floppy disk is made of thin plastic, coated with a magnetic recording surface rather like that used on tape. The disk, in its protective envelope, is placed in a **disk drive** which rotates it and moves a read/write head across the disk's surface. The disk is divided into concentric rings called **tracks**, each of which is in turn subdivided into **sectors**. Using a program called a **disk operating system**, the computer keeps track of exactly where information is on the disk and it can get to any item of data by moving the head to the appropriate track and then waiting for the right sector to come round. Two methods are used to tell the computer where on a track each sector starts: **soft sectoring** where special signals are recorded on the surface and **hard sectoring** where holes are punched through the disk around the central hole, one per sector.

Half-way between cassettes and disks is the **stringy floppy** — a miniature continuous loop tape cartridge, faster than a cassette but cheaper than a disk system. **Hard disk** systems are also available for microcomputers; they store more information than floppy disks, are more reliable and information can be transferred to and from them much more quickly.

You, the user, must be able to communicate with the computer and the generally accepted minimum for this is the **visual display unit (VDU)**, which looks like a TV screen with a typewriter-style keyboard; sometimes these are built into the system, sometimes they're separate. If you want a written record (hard copy) of the computer's output, you'll need a **printer**.

The computer can send out and receive information in two forms — **parallel** and **serial**. Parallel input/output (I/O) requires a series of wires to connect the computer to another device, such as a printer, and it sends out data a byte at a time, with a separate wire carrying each bit. Serial I/O involves sending data one bit at a time along a single piece of wire, with extra bits added to tell the receiving device when a byte is about to start and when it has finished. The speed that data is transmitted is referred to as the **baud rate** and, very roughly, the baud rate divided by 10 equals the number of bytes being sent per second.

To ensure that both receiver and transmitter link up without any electrical horrors, standards exist for serial interfaces; the most common is **RS232** (or **V24**) while, for parallel interfaces to printers, the **Centronics** standard is popular.

Finally, a **modem** connects a computer, via a serial interface, to the telephone system allowing two computers with modems to exchange information. A modem must be wired into the telephone system and you need Telecom's permission; instead you could use an **acoustic coupler**, which has two obscene-looking rubber cups into which the handset fits, and which has no electrical connection with the phone system — Telecom isn't so uppity about the use of these.

Australia's first under $300 COMPUTER...

$295
INCL. ZX80 BASIC MANUAL

Remember — *all prices shown include sales tax, postage and packing.*
N.B. Your Sinclair ZX80 may qualify as a business expense.

sinclair ZX80 -British made.

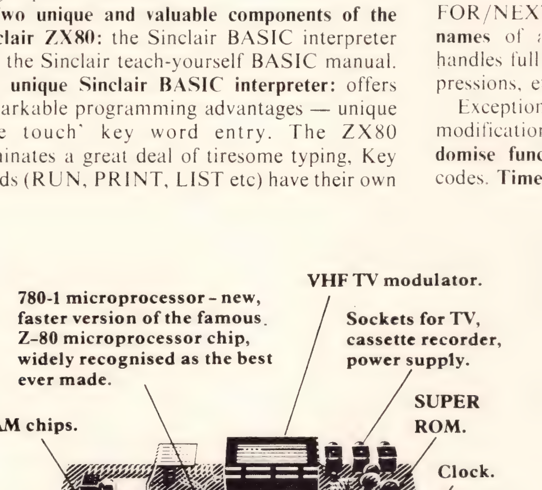
Until now, building your own computer could cost you around $600 — and still leave you with only a bare board for your trouble. The Sinclair ZX80 changes all that. For just $295 you get **everything** you need including leads for direct connection to your own cassette recorder and television. The ZX80 really is a complete, powerful full-facility computer matching or surpassing other personal computers costing much more. The ZX80 is programmed in BASIC and you could use it for anything from chess to running a power station.

Two unique and valuable components of the Sinclair ZX80: the Sinclair BASIC interpreter and the Sinclair teach-yourself BASIC manual. **The unique Sinclair BASIC interpreter:** offers remarkable programming advantages — unique 'one touch' key word entry. The ZX80 eliminates a great deal of tiresome typing. Key words (RUN, PRINT, LIST etc) have their own single key entry. Unique syntax check. Only lines with correct syntax are accepted into programs. A cursor identifies errors immediately, preventing entry of long and complicated programs with faults only to discover them when you run.

Excellent string handling capability — takes up to 26 string variables of any length. All strings can undergo all rational tests (e.g. comparison). The ZX80 also has string input to request a line of text; strings do **not** need to be dimensioned. Up to 26 single dimension arrays. FOR/NEXT loops nested up to 26. **Variable names** of any length. BASIC language also handles full Boolean arithmatic, conditional expressions, etc.

Exceptionally powerful edit facilities, allows modification of existing program lines. **Randomise function,** useful for games and secret codes. **Timer under program control.** PEEK and POKE enable entry of machine code instructions. USR causes jump to a user's machine language sub-routine. **High resolution graphics** with 22 standard graphic symbols. **The Sinclair teach-yourself-BASIC manual** 96 page book **free** with every kit.

Fewer chips, compact design, volume production means MORE POWER FOR YOUR DOLLAR! The ZX80 owes its low price to its remarkable design: the whole system is packed onto fewer, newer more powerful and advanced LSI chips. A single SUPER ROM, for instance, contains the BASIC interpreter, the character set, operating system and monitor. And the ZX80's 1K byte RAM is roughly equivalent to 4K bytes in a conventional computer because the ZX80's brilliant design packs the RAM so much more tightly. (Key words occupy just a single byte). You can add to the memory via the expansion port, giving a maximum potential of 16K.



ORDER FORM: SINCLAIR EQUIPMENT (AUSTRALASIA) PTY. LTD.
308 High St., Kew 3101, Vic. Tel. 861 6224.

Quantity	Item	Item Price	Total
	Ready-assembled Sinclair ZX80 Personal Computer(s). Price incl. ZX80 BASIC manual, excl. mains adaptor.	$295.00	
	Mains Adaptor(s) (600Ma at 9V DC nominal unregulated).	$16.00	
	Memory Expansion Board(s) takes up to 3K bytes.	$28.50	
	RAM Memory chips — standard•5K bytes capacity.	$10.00	
	Sinclair ZX80 Manual(s) free with every ZX80 computer.	$15.00	

I enclose cheque/Bankcard/Diners Club/Amex TOTAL

Name ____________________

Address ____________________ Postcode ________

MICROCHESS

The latter half of 1980 saw three major computer chess tournaments, two of which were restricted to micros. APC's resident chess expert Kevin O'Connell examines the results.

The first World Microcomputer Championship was treated with the full respect it deserved, being held under the auspices not only of the ICCA (International Computer Chess Association) but also under those of FIDE (Federation Internationale des Echecs) — the World Chess Federation. Held in London, as part of the Personal Computer World Show (4-6 September), it was certainly a prestigious event, which the commercial entries, seeking publicity from the World Champion title for their program, proved by sending top executives to oversee the operation of their machines.

There were 14 competitors — there could have been more but quite a few enquiries arrived too late; a very strong American program arrived two days after the tournament ended! Originally sent in June with a 15 cent stamp, it had been returned to sender because of insufficient postage, was put back in the mail bearing a second 15 cent stamp (just one cent short of the airmail tariff) and so travelled by sea, no doubt leaving the programmer feeling rather sick.

The first round results would have completely justified the seeding had not Sargon 2.0 lost on time when it was more than a queen ahead on material. K. Chess IV also had an unfortunate experience — the draw gave it the white pieces but it could only play with black! K. Chess IV lost that game by default, but some hurried modifications enabled it to complete the course.

Round two put a check to the aspirations of Mike Johnson, winner of the 1978 Personal Computer World tournament, and David Broughton, winner of last year's non-commercial prize. David Broughton's Vega went down to Sargon 2.0 while Mike 3.0, the only program running on special chess hardware, drew by repetition against the Modular Game System 2.5 despite being a queen up.

The two programs that had now emerged as favourites met in the third round. The destiny of the non-commercial first prize seemed almost sure to go to Mike Johnson and Dave Wilson when Mike 3.0 won while Vega lost again.

In round five, Chess Challenger played its most convincing game of the Championship, beating the Modular Game System 2.5. Since Challenger's other nearest rival at the start of this round, Mike 3.0, also lost, Challenger's lead, with just one round to go, extended to a full point.

In the last round attention was focused on the games Sargon 2.0 v Chess Challenger and Rook 4.0 v Mike 3.0, with a weather eye cast on developments in Boris Experimental's game. If Sargon 2.0 could beat Challenger and Boris Experimental could defeat its opponent (which it did), there would be a tie for first place and the excitement of a play-off match.

The game between Rook and Mike would determine the winner for the top non-commercial prize. In the event these two programs drew and there could have been a tie for all three non-commercial prizes had Vega been able to win, but it was having an uphill struggle to draw with the Auto Response Board 2.5.

Mike 3.0 and Rook 4.0 shared $1500 for the top two non-commercial entries, the other prize of $200 going to Vega 1.7. Chess Challenger took the handsome trophy and the glory of being the first program to bear the illustrious title of World Microcomputer Chess Champion.

Looking through the final score-table it seems almost to be a match tournament Dan and Kathe Spracklen against the rest of the world, an event they lost by the narrowest of margins; the five programs written by them (entirely or in large part) scored 17 out of the total available 35 points.

Table of results

PROGRAM (* commercial entry)	R1	R2	R3	R4	R5	Tot.	S/ded
1 CHESS CHALLENGER * (USA) (Dan & Kathe Spracklen, Ron Nelson, Frank Duason & Ed English) (6502 - Assembler) 20 k	W12	W10	W2	W7	W5	5	5
2 BORIS EXPERIMENTAL* (USA) (programmers not named but based on Boris 2.5 by Dan & Kathe Spracklen) (6502 - Assembler) 8 k	W14	W8	L1	W3	W7	4	6
3 MIKE 3.0 (UK) Mike Johnson & Dave Wilson (6502 & chess hardware - Assembler) 48 k	W10	D7	W11	L2	D4	3	3
4 ROOK 4.0 (Sweden) Lars Kalsson (Z8000 - Assembler) 16 k	L9	D12	W10	W6	D3	3	9
5 SARGON 2.0* (USA) Dan & Kathe Spracklen (6502 - Machine Language) 24 k	L11	W9	W13	W12	L1	3	7
6 GAMBIET* (Netherlands) Wim Rens (Z80 - Assembler) 10k	L7	W11	W8	L4	W12	3	11
7 MODULAR GAME SYSTEM 2.5* (USA) Dan & Kathe Spracklen (6502 - Assembler) 8 k	W6	D3	W9	L1	L2	2½	4
8 AUTO RESPONSE BOARD 2.5* (USA) Dan & Kathe Spracklen (6502 - Machine Language) 8 k	W13	L2	L6	W11	D9	2½	1
9 VEGA 1.7 (UK) David Broughton (Z80 - Assembler) 12 k	W4	L5	L7	W13	D8	2½	2
10 VIKTOR (Switzerland) Herbert Bruderer (8085 - Assembler) 8 k	L3	L1	L4	W14	W13	2	10
11 ALBATROSS (UK) Michael Parker (Z80 - Assembler) 18 k	W5	L6	L3	L8	W14	2	14
12 FAFNER 2 (UK) Guy Burkill & Alex Kidson (6502 - Pascal & Assembler) 16k	L1	D4	W14	D5	D6	1½	12
13 PRINCHESS 1.0 (Sweden) Ulf Rathsman (6502 - Assembler) 12 k	L8	W14	L5	L9	L10	1	8
14 K. CHESS IV (UK) Andrew Thomason (Z80 - Machine Language) 2.2 k	L2	L13	L12	L10	L11	0	13

MICROCHESS

US championships

No sooner had the dust settled on the World Microcomputer Chess Championship at the Personal Computer World Show than the results of the first US Microcomputer Chess Championship, played that same weekend, came through.

The North American tournament was played in San Jose, California, and co-sponsored by *Personal Computing*, the Mychess team, Applied Concepts and Motorola. Three of the competitors were also present in London at the same time — one of the advantages a micro has over a human.

The results achieved by Boris 2.5 and Chess Champion Super System III were particularly impressive — they have been commercially available for more than a year now. Of course, this is not to detract from the success of the Spracklens' new program in the Challenger hardware, but it will be interesting to see the new programs expected from Applied Concepts and SciSys.

US results

1 (1)	CHAMPION SENSORY CHALLENGER (USA) Stand-alone device	W6	W5	W2	W4	4
2= (2)	BORIS 2.5 (USA) Stand-alone device	W10	W7	L1	D5	2½
2= (3)	BORIS EXPERIMENTAL (USA) Stand-alone device	W8	D4	L5	W9	2½
2= (4)	CHESS CHAMPION SUPER SYSTEM III (HK) Stand-alone device	W7	D3	W8	L1	2½
2= (5)	MYCHESS 'B' (USA) Cromemco	W9	L1	W3	D2	2½
6 (6)	ATARI 4k 'A' (USA) TV Interface Unit	L1	W10	D7	D8	2
7= (7)	MYCHESS 'A' (USA) Cromemco	L4	L2	D6	W10	1½
7= (8)	SFINKS (USA) TRS-80	L3	W9	L4	D6	1½
9 (9)	ATARI 4k 'B' (USA) TV Interface Unit	L5	L8	W10	L3	1
10 (10)	LANE'S TC'86 (USA) Stand-alone device	L2	L6	L9	L7	0

World computer chess championship

After their own two special tournaments, the big guns of the micro world were trained on Linz in Austria, site of the third World Computer Chess Championship (25-29 September). Four 'micros' were among the 18 contestants. Advance 1.0 (UK) and Bebe (USA) were under microprocessor control but also availed themselves of some special chess hardware which, in each case, owed much to bit-slice technology. There were also two pure micros: Mychess (USA), running on a Cromemco, and the ubiquitous Champion Sensory Challenger (USA), victor of both the US and the World Microcomputer Championships.

Advance 1.0, programmed by Mike Johnson and Dave Wilson, played in the World Micro Championship under the guise of Mike 3.0. In Linz only the name (and program!) had been changed. By defeating Sweden's Dark Horse, running on a Univac 1100/81 and programmed by Ulf Rathsman — whose Princhess had competed in the World Micro where it had lost to Mike 3.0 — this was the only microcomputer to win a game. We reproduce that game here:

WHITE: Advance 1.0
BLACK: Dark Horse:1.0

1 e2-e4 e7-e5
2 Ng1-f3 Nb8-c6
3 Bf1-b5 Ng8-f6
4 0-0 (Ke1-g1) Bf8-e7
5 Rf1-e1 Be7-d6??

All beginners know that it is bad to move a piece twice in the opening.

6 d2-d4 Nc6xd4
7 Nf3xd4 a7-a6

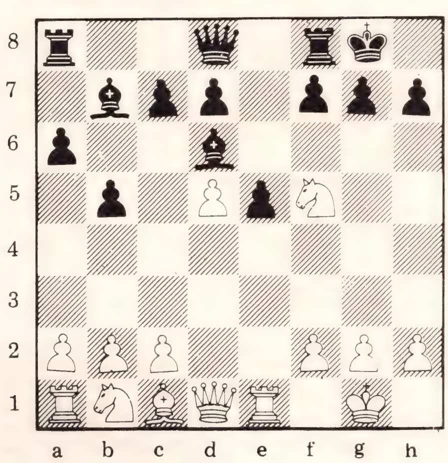
Why not? After all, if White has two pieces attacked then Black must be able to take one of them, so there is no rush to capture on d4.

8 Bb5-c4 b7-b5

White still has two pieces attacked.

9 Bc4-d5 Nf6xd5
10 e4xd5

Now Black wants, and needs, to take the knight on d4. That is what it was counting on, but White's last move cleared one more piece from the e-file and now e5xd4 is illegal. Oh well, back to the keyboard!



10 ... 0-0 (Ke8-g8)
11 Nd4-f5 Bc8-b7
12 Bc1-h6! g7xh6?

Black was lost, but this makes matters worse.

13 Qd1-g4+ Black resigned

The trouble with 13...Qd8-g5 would have been that after 14 Nf5xh6+, Black would be at something of a loss to counter the follow-up 15 Qg4xg5.

The overall performance of the micros in Linz was quite good — of 14 games played against programs running on mainframes the micros scored +1 =7 —6 (32%).

See the score table for the full results of the tournament.

World CCC results

1.	BELLE (USA) Ken Thompson; Joe Condon PDP 11/70 with chess hardware	W8	D6	W4	W3	3½
2.	CHAOS (USA) Fred Swartz; Mike Alexander; Jack O'Keefe; Victor Berman Amdahl 470	W17	D4	W6	W7	3½
3.	DUCHESS (USA) Tom Truscott; Bruce Wright; Eric Jensen Amdahl V/8	W18	W12	W7	L1	3
4.	L'EXCENTRIQUE (Can) Claude Jarry Amdahl V/7	W5	D2	L1	W12	2½
5.	CHESS 4.9 (USA) Lawrence Atkin; David Cahlander CDC Cyber 176	L4	D9	W11	W10	2½
6.	NUCHESS (USA) David Slate; William Blanchard CDC Cyber 176	W16	D1	L2	D9	2
7.	KAISSA (USSR) V L Arlazarov; M V Donskoy IBM 370/168	W14	W13	L3	L2	2
8.	BCP (UK) Don Beal PDP 11/70	L1	D10	D14	W15	2
9.	BEBE (USA) Tony Scherzer Bebe chess machine	D10	D5	D12	D6	2
10.	SCHACH 2.3 (BRD) Matthias Engelbach Burroughs 7800	D9	D8	W13	L5	2
11.	AWIT (Can) T A Marsland Amdahl 470 V/7	L13	W17	L5	W18	2
12.	MASTER (UK) Peter Kent; John Birmingham IBM 3033	W15	L3	D9	L4	1½
13.	OSTRICH (Can) Monroe Newborn Data General Nova 4	W11	L7	L10	D14	1½
14.	MYCHESS (USA) David Kittinger Cromemco	L7	D15	D8	D13	1½
15.	PARWELL (BRD) Thomas Nitsche; Elmar Henne; Wolfram Wolff Siemens SMS 2 plus 128 8080s in parallel	L12	D14	W16	L8	1½
16.	ADVANCE 1.0 (UK) Mike Johnson; Dave Wilson 6502 plus chess hardware	L6	D18	L15	W17	1½
17.	DARK HORSE (SWE) Ulf Rathsman Univac 1100/81	L2	L11	W18	L16	1
18.	CSC (USA) Dan & Kathe Spracklen; Ron Nelson; Frank Duason; Ed English Champion Sensory Challenger	L3	D16	L17	L11	½

Belle defeated Chaos in the play-off for the title

EXCITING, ENTERTAINING SOFTWARE FOR YOUR SORCERER

If you enjoyed 'INVADERS', you'll love GALAXIANS by Martin Sevior.
Your space ship has encountered an army of Galaxians whose mission is to destroy you. Blast them with lazers, but beware, they dive at you with ever increasing fury! Your Sorcerer's unique high resolution graphics add to the arcade-like excitement that this game generates. Program comes on cassette and will run on any size Sorcerer.

$19.95

System Software

1 Kent Street Bicton, Western Australia 6157 Telephone (09) 339 3842

WHERE TO GET IT:

Write to the above address. We will accept money orders and personal or bank cheques. Add $2.00 for handling and airmail postage.
Dealer enquiries invited.

COMPUTER GAMES

by David Levy

GUESSING THE ODDS

Deduced probabilities

When playing a game of cards you usually know which cards you have been given, but normally you will not see the cards that have been dealt to your opponent(s). You may be able to deduce certain things about an opponent's card holding from the way in which he bids or plays, but it is unlikely that you will know exactly what he holds until very near the end of the hand. Decisions made in this sort of environment must be made on a probabilistic basis; in other words, you play with the odds and hope for the best. If you have calculated the odds correctly you will win more often than you lose.

Shuffling

Before proceeding to the main point of this month's article I should perhaps interpose a brief section on how to shuffle the cards in your program. The simplest way of creating a randomly sorted deck is as follows. Starting with the deck in any order you wish (even perfectly sorted), interchange the first card in the deck with the Rth card, where R is a pseudo-randomly chosen integer on the range 1 to n (n is the total number of cards in the deck). Then interchange the second card with another randomly chosen card, then the third, and so on to the end of the pack. The manner in which you generate your random numbers is of some consequence — I would recommend that while developing your program you use one of the seeding methods in which the i+1 th random number is generated from the i th number, and the series is started with a "seed" which may be chosen by the user. This approach has the advantage that if you spot a bug in your program you can recreate the hand simply by starting with the same seed. Once your program is debugged you may use the computer's internal clock to supply the seed, for example by using the time elapsed between the pressing of two keys.

One seeded random number generator which will suffice is:

$R_i = a^i \times \text{seed} \pmod m$

where R_i = i th pseudo-random number

$a = 8t + 3$ (for any positive integer t)

$m = 2^b$ where b is the number of bits per word in your computer.

Deducing information from the Play of the Cards

For the purpose of creating a simple example I have invented the following card game. The game is played by three players who are each dealt 17 cards at the start of a hand. The 52nd card in the deck is turned face up and that suit is trumps.

Starting with the player on the dealer's left, the player leads a card and the other players must follow suit if they can, or they may trump if they wish (provided that they are unable to follow suit). The player who wins one trick leads to the next, and the player who wins most tricks wins the hand.

Let us assume that we are dealt the following hand:

Table 1	A	K	Q	J	10	9	8	7	6	5	4	3	2
SPADES:	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.0	0.5	0.5	0.0	0.5	0.0
HEARTS:	0.5	0.5	0.0	0.5	0.0	0.5	0.5	0.0	0.5	0.0	0.5	0.5	0.5
DIAMONDS:	0.5	0.0	0.5	0.5	0.0	0.0	0.5	0.5	0.0	0.5	0.5	0.5	0.0
CLUBS:	0.5	0.5	0.5	0.0	0.5	0.5	0.0	0.5	0.0	0.5	0.0	0.5	0.5

SPADES (trumps): A K 4 2
HEARTS: Q 10 7 5
DIAMONDS: K 10 9 6 2
CLUBS: J 8 6 4

and that the 7 of spades is the card turned up. It is our turn to lead first.

At the start of the hand we know absolutely nothing about which cards our opponents hold, except for the fact that between them they hold all 34 of the unseen cards. But we do not have any indication as to how these 34 cards are distributed between the unseen hands, so the probability of each of the cards being in a particular hand is 0.5. We can therefore begin to construct, for each of our opponents (Bill and John) probability estimates for each card in the deck. At the start of the hand the estimates for each of them will be as shown in Table 1.

Assume that we lead the 4 of spades, and that the next player (Bill) plays the 9 of spades and the third player (John) takes the trick with the Q. What have we learned about the probabilities of the other cards, if anything?

Before answering this question I must explain an important theorem from Probability Theory, called Bayes' theorem.

Bayes' theorem

Let us suppose that there are two bags, each containing five balls. Bag A contains 1 white and 4 black balls, bag B contains 3 white and 2 black balls. I take a ball at random from one of the bags, and the ball is white. What is the probability that I took the ball from bag A?

The probability that a ball selected at random from bag A will be white is 1/5.

The probability that a ball selected at random from bag B will be white is 3/5.

Bayes' theorem shows that the probability that a randomly selected white ball actually came from bag A =

$$\frac{1/5}{(1/5 + 3/5)} = 1/4$$

The reader will be able to generalize from this example, and the application to our game of cards will soon become apparent.

What have we learned?

Let us now return to the question of what, if anything, we have learned about Bill and John's hands from the cards they played to trick one? We probably cannot say very much at all about Bill's hand at the moment, but we already know something about John's cards.

John took the first trick with the Q of spades. The A and K are in our own hand and so the only cards that John could possibly have used to take the trick were the Q, J and 10. If John had held the Q and 10 but been missing the J, he would have played the 10, so from the fact that he played the Q we know that his original spade holding included:

Q, J and 10 *or* Q and J *or* Q (without J or 10).

Now we can use the tables of probabilities for the individual cards to determine the *a priori* probability that John held each of these three holdings:

Probability that he held the Q, J and 10 = 0.5 x 0.5 x 0.5 = 0.125
Probability that he held the Q and J but not the 10 = 0.5 x 0.5 x 0.5 = 0.125 (Note that since the probability of his holding the 10 is 0.5, the probability of his not holding it is 1—0.5=0.5)
Probability that he held the Q but not the J or 10 = 0.5 x 0.5 x 0.5 = 0.125

And from Bayes' theorem we can show that the probability that the Q came from each of these three holdings is:

Q,J,10: 0.125/(0.125+0.125+0.125)=1/3
Q,J: 0.125/(0.125+0.125+0.125)=1/3
Q: 0.125/(0.125+0.125+0.125)=1/3

Note that had the calculations been performed later in the hand, when the probabilities were not all equal (0.5), the final values would not all have been 1/3.

From these last calculations we can see that the probability that John holds the 10 of spades is 1/3 (in which case he also holds the J), and the probability that he holds the J is 2/3. We can therefore adjust the probabilities for the individual cards in John's hand as follows:

For the 10 of spades: probability=0.333
For the J of spades: probability=0.667
For all other unseen cards the probabilities are equal, and these are:

$$\frac{16-0.333-0.667}{32-1-1} = \frac{15}{30} = 0.5$$

Since there are 16 unseen cards in John's hand, and 32 unseen cards in total (the probabilities of the J and 10 of spades being in John's hand are subtracted from the number of cards in his hand, and one is subtracted for each of them from the total number of unseen cards).

If the probability of the J of spades being in John's hand is 0.667, then the probability of it being in Bill's hand is 0.333, and by the same argument the probability of Bill holding the 10 of spades is 0.667. So we have been able to make some adjustments in the probabilities simply on the basis of John having played the Q of spades at trick one. We can also make note of the fact that if John ever shows the 10 of spades, we will know that he holds the J.

At trick two, John must lead because he won trick one. He leads the A of hearts, we play the 5, and Bill trumps with the 8 of spades. What have we learned from trick two? First of all, Bill would obviously use his lowest trump or one of his lowest contiguous group of trumps. The 7 was the original face up card, we played the 4 on trick one and Bill played the 9. We hold the 2 of spades and so Bill's 8 of spades must have been played from one of the following holdings:

J,10,8,6,5,3:
J,10,8,6,5:
J,10,8,6:
J,10,8:
10,8,6,5,3:
10,8,6,5:
10,8,6:
10,8:
8,6,5,3:
8,6,5:
8,6:
8:

and by using Bayes' theorem we can determine the probabilities of each of the above cards being in Bill's hand, and from these probability estimates we can determine estimates for the cards being in John's hand. We can also adjust the probabilities for all the hearts: those which are not in our own hand must all be in John's hand.

Deducing information from the bidding

In many card games there is a bidding phase between the deal and the play of the cards. The best known of such games is Bridge, but the popular German game of Skat is another widespread example (it is said that Skat can be played by more than 50% of the entire population of Germany). Since each bid has a meaning, it should be possible for the card playing program to learn something about its opponents' hands from the way that they bid, and it can then adjust its probability estimates for each card in their hand. How this is done will obviously vary from one game to another. Let us take a brief look at Bridge, to see how we might modify the probability estimates of the unseen cards in the light of the bidding.

We are sitting South and hold 10 high card points. We look at the 13 cards in our hand and assign a probability of 1/3 to each of the reamining 39 cards in each of the other three hands. West opens the bidding and bids one spade, indicating that he has a stronger than average hand and that spades is his best suit. (Of course, this bid can mean other things, but we shall assume for this example that the above meaning is correct in the particular bidding system that West and his partner employ.) We may now adjust the probabilities of the spades, so that each spade in West's probability array has a slightly higher probability (say 0.45 instead of 0.33), and we may also adjust the probabilities of the high valued cards (aces, kings, queens and jacks) so that they give an expected high card holding which corresponds to a typical one spade opening bid. (If this bid is made with an average of 13 points, the ace counting 4 points, king 3, queen 2 and jack 1, then by making each of the high card probabilities 0.4333 we give West an expectation of 13 out of the remaining high card points: there are 40 high card points in total and we hold 10 of them, leaving 30, and 13/30=0.433). We should, in fact, give a slightly higher probability to a card which is both a spade and a high card.

Having assigned new probabilities to

the spades and the high cards, we can then adjust the probabilities for the remaining cards in West's hand, so that the sum total of his probabilities is 13 (the total number of cards in his hand), and we can adjust the probabilities for each card in North's and East's hands by subtracting West's probability from 1 and dividing the result by 2, remembering to ignore all cards in our own hand.

So from his first bid we can make quite a lot of probabilisitic estimates about West's cards, and hence about those in the North and East hands.

The bidding then passes to North, and depending on his bid we make adjustments to his probabilities using similar, logical arguments, and then we adjust the probabilities for West and East. This process continues until the end of the bidding — each time we acquire some information that increases the likelihood of a card being in a particular place, we increase the probability for that place and reduce it accordingly in the other hands. When there is some negative information about the position of a card we use it in a similar way.

By the end of the bidding phase a good bridge program should have a fairly accurate estimate of how each of the other three hands is made up. By summing the probability values for all the spades in a hand the program can get an estimate of how many spades that player holds. By summing the products of the high card probability x high card point values, the program can estimate the number of high card points in each suit in each hand. It will then be better able to plan its play of the hand, and of course the probabilities will be adjusted all through the playing phase.

How to use deduced information

The most obvious use of our deduced probabilities arises when the probability estimates for all unseen cards are all either 1 or 0, i.e. we know where all the remaining cards lie. We then have a case of a perfect information game, and we can solve this game by performing a tree search to the end of the game. Even though there may be three or more players, the tree approach should still work, though we must make certain assumptions about the way that the other players are going to make their decisions. For example, let us assume that we are two tricks from the end of a hand of our three-player card game.

We hold: A of spades, 10 of diamonds.
Bill holds: J of diamonds, 5 of clubs.
John holds: 3 and 2 of clubs.
It is our turn to lead (remember that spades are trumps).

The program now constructs a game tree, of depth 6-ply. Part of the tree will look like the above diagram. We assign to the terminal nodes of the tree, scores corresponding to the number of tricks won by each player, and we back-up through the tree until we can determine which card should be played next. In this example the situation is simple because if we lead the A of spades first we *may* take two tricks, whereas if we lead the 10 of diamonds we can only make one trick. Note the use of the word "may". In order to make two tricks we need some help from Bill, who must make a mistake and discard the J of diamonds in the hope that our second card is the 2 or 3 of clubs and he will make his 5. But since we lose nothing by playing the A of spades first, that is clearly the best way to continue. How can we modify our traditional methods of tree-searching to cater for situations such as this one, in which we wish to allow for the possibility that our opponent will make a mistake? Fortunately the problem has been solved for us, by the ubiquitous Donald Michie, whose name crops up time and again in interesting research reports on various topics within the science of Artificial Intelligence.

Expected values in backed-up trees

Michie's method, which I shall discuss in some detail in a later article, is based on the assertion that when searching a game tree it is unreasonable to assume perfect play by the opponent, since there must always be a finite chance that he will not choose the best move. Let us see how this helps us to search the above tree.

We may simplify Michie's concept as follows: If there is a 99% chance that Bill will play the 5 of clubs from position P_1, and a 1% chance that he will play the J of diamonds, then since the 5 of clubs will give us a score of 1 (i.e. we take one trick) and the J of diamonds will give us a score of 2, the expected value to us of position P_1 is

$$(0.99 \times 1)+(0.01 \times 2)=0.99+0.02=1.01$$

whereas if we play the 10 of diamonds from position P_0, the expected (in fact the certain) value of position P_2 will be 1 (i.e. we will take one trick no matter how Bill and John play). Since 1.01 is greater than 1, we should play the A of spades from P_0 because it maximises our expected score. The reader will probably have realised by now that not only does Michie's method allow us to optimize our practical chances when we *know* exactly where all the unseen cards lie, it also enables us to use our probability estimates of the locations of the unseen cards, to build game trees which will help in the play of the hand. In other words, Michie has shown us how to play with the odds!

Task for the month

Find or invent a simple card game in which information may be deduced from the play of the cards. (Avoid bidding games, unless you are extremely confident and have many free hours this month.) Write a program to play this game, modifying the probability estimates of the unseen cards in the light of the user's play. Experiment with various methods of adjusting these estimates until the program plays at least moderately sensibly. At the point in the game where exhaustive search will not be too time consuming, set up a probabilistic game tree à la Michie to search to the end of the game.

Bibliography

Michie, D.: *A theory of evaluative comments in chess.* Memorandum MIP-R-105, Department of Machine Intelligence and Perception, University of Edinburgh, July 1974.

Mirhram, G.A.: *Pseudo-Random Number Generators are really Card Shufflers.* Personal Computer Proceedings, National Computer Conference, New York, 1979, pp. 318-326.

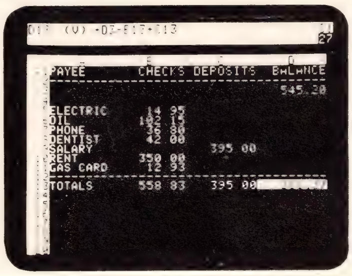
(For more information on Bayes' theorem see any good book on statistics or probability theory.)

NEW Have we got a

VISICALC NOW FOR TRS80

Students, professors, small businessmen, everyone raves about **VISICALC.** Permits you to dispense with pencil, paper and calculator — instea you have an 'electronic sheet' to do your calculations, projections and planning. You can calculate sales projections, income taxes, personal budget, cost estimates, engineering changes — even balance your own chequebook.

VISICALC is partitioned into a matrix of rows and columns. At each position in this matrix you can define a title, formula or number. By writing on your 'electronic sheet', you can set up individualised charts, tables and records.



VISICALC can recalculate any problem. Suppose you change a number you had previously 'written on the 'electronic sheet'. Like magic, the computer will remember the original formulas and calcula- tions you've used and all the other related numbers on the sheet change before your eyes. **VISICAL** automatically recalculates all the relevant formulas. Recalculation makes **VISICALC** a powerful planning and forecasting tool. Not only can you effortlessly correct mistakes and omissions, you ca also examine various alternatives.

For example, imagine you are doing sales projections using **VISICALC.** You may want to know wh the impact on your company will be if a specific product doesn't sell as well as you anticipate. Wha you sell only 200 a month instead of 250? What if you sell 300? What if one of your salesmen quits and it takes his replacement six weeks to catch up? With **VISICALC** to assist you in performing the calculations, you can save countless hours.

It has been said that **VISICALC** alone justifies the purchase of a microcomputer!

An invaluable piece of software for everyone . . . **TRS80 32K disk only $195.**
Also available for **APPLE II 32K with one disk . . . $195**

COMPUKIT 101

IF YOU WANT TO LEARN ABOUT COMPUTERS, BUT DIDN'T KNOW WHICH MACHINE TO BUY, THEN THIS IS THE MACHINE FOR YOU.

The **COMPUKIT 101** is a real computer — start with 4K RAM memory and full 8K Microsoft Basic in ROM. The **COMPUKIT 101** connects directly to your TV — just assemble, hit 'RETURN' and GO! A cassette interface allows you to use a normal cassette recorder to save and load programs. Because the **COMPUKIT 101** is available only as a kit, the savings are enormous.

- o 6502 based system — best value for money on the mark
- o Powerful 8K Basic — fastest around
- o Full Qwerty Keyboard
- o 4K RAM expandable to 8K on board
- o Power supply and RF modulator on board
- o No extras needed — Plug-in and go
- o Kansas City Tape Interface on board
- o Free Sampler tape including powerful Dissassembler an Monitor with each Kit.

COMPUKIT 101 kit, including all parts, manual, RF modulator and power supply	**$595.**
Additional 4K RAM Memory (fits on board)	**85.**
Assembler/Editor tape	**25.**
Screen Editor tape	**6.**
Space Invaders (8K)	**14.**

ADVENTURE

ADVENTURE by Scott Adams is like no other program you have ever seen. "Out of 50 programs reviewed, **ADVENTURE** was rated No. 1 . . . highly recommended" —'80 Software critique.

Defeat exotic wild animals to get treasures or work out how to get out of a quicksand bog — nine different adventures now available.

1. **ADVENTURELAND** — you wander through an enchanted world trying to recover the 13 lost treasures. You'll encounter wild animals, magical beings and many other perils and puzzles.
2. **PIRATES ADVENTURE** — you'll meet up with the pirate and his daffy bird along with many other strange sights as you attempt to go from your flat to Treasure Island. Can you recover Long John Silver's lost treasures?
3. **MISSION IMPOSSIBLE ADVENTURE** — hello, your mission is to . . . and so it starts. Will you be able to complete your mission in time? Or is the world's first nuclear reactor doomed? This one is well named. It's hard, there is no magic, but plenty of suspense.
4. **VOODOO CASTLE** — Count Cristo has had a fiendish curse put on him by his enemies. There he lies with you his only hope. Will you be able to rescue him or is he doomed forever?
5. **THE COUNT** — you wake up in a large brass bed in a castle somewhere in Transylvania. Who are you? What are you doing here and WHY did the postman deliver a bottle of blood?
6. **STRANGE ODYSSEY** — marooned at the edge of the Galaxy, you've stumbled on the ruins of an ancient alien civilization complete with fabulous treasures and unearthly technologies. Can you collect the treasures and return or will you end up marooned forever?
7. **MYSTERY FUNHOUSE** — can you find your way completely through the strangest Fun House in existence or will you always be kicked out when the park closes?
8. **PYRAMID OF DOOM** — an Egyptian treasure hunt leads you into the dark recesses of a recently uncovered Pyramid. Will you recover all the treasures or will you join its denizens for their long eternal sleep?
9. **GHOST TOWN** — explore a deserted mining town in search of 13 treasures from rattlesnakes to runaway horses — this adventure's got them all! (Also includes new bonus scoring system!)

TRS80 16K — $19.50 each
APPLE II 24K — $19.50 each
SORCERER 16K — $19.50 each
SYSTEM 80 16K — $19.50 each

JUST RELEASED:
Can't figure out how to get the ox out of the bog or where the Pharoah's heart is? These and many other clues are given in a special format actually designed to help you as little as possible but still let you solve the ADVENTURE.

ADVENTURE HINT BOOK (1-9) only $12.50

program for you!

NEW

ASTEROID NOVA

If you liked 'Invaders', you will love **ASTEROID NOVA.** Finally **ASTEROIDS** come to micros: Your spaceship is travelling in the middle of a shower of asteroids. Your laser is your only defense against the asteroids. Blast them but beware — big asteroids fragment into small asteroids and you are surrounded! From time to time you will encounter an alien spaceship whose mission is to destroy you, so you'd better destroy it first! High resolution graphics and super sound effects.

APPLE II 32K disk $29.50
TRS80 Level II 16K tape $19.50

SUPER INVADERS

The video game that swept the world now comes into your own home — original graphics and sound: 55 aliens advance and shower you with lethal arrows; as you shoot them they quicken their descent. Written in machine language for rapid action these are the best versions of the Space Invaders video game: totally graphic and twice the speed of Alien Invaders; includes sound!!

TRS80 16K — $24.50
APPLE II 32K — $29.50

:DITOR/ \SSEMBLER PLUS

ow get every feature of Editor/Assembler d T-Bug all in one package **PLUS** — many ig computer' features to simplify your sembly language programming and de- gging. Don't waste time creating both urce and object tapes — **ASSEMBLER .US** assembles directly into memory. acros are supported: simply define a acro and frequently used sequences are tomatically assembled. Seven new ex- essions. Improved symbol printout and a ash command to give you more storage ace for general text editing! The Editor is proved with commands like Substitute, ove, Copy and Extend, for moving blocks d repeating them anywhere for searching d substituting occurrences in the text. A nge command is available for both Edit d Find enabling you to operate on a single e or a range of lines. The debugger, Bug, allows up to eight breakpoints, gle-stepping, etc.

)ITOR ASSEMBLER PLUS comes with eighty page manual.

{S80 16K tape $39.50

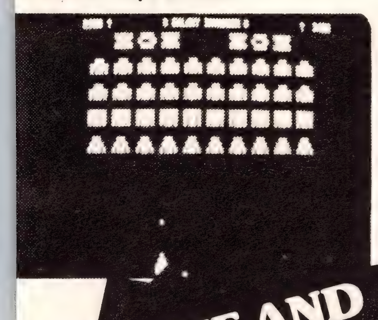
GALAXY INVASION

he newest and most exciting Invaders- ype game yet! Turn your **TRS80** to a video game arcade. **GALAXY NVASION** is based on **MOON ALIEN** a phalanx of space invaders above the irth: you are the sole defender. As you re your laser at the invaders, they voop down and bomb you, but there e no bases to hide behind!

icredibly smooth graphics and great und.

ighly recommended

RS80 16K Tape $19.50



COME AND SEE US

FOR THE SINCLAIR ZX80

'30 PROGRAMS FOR THE SINCLAIR ZX-80: 1K' is a unique 112 page book which contains 30 programs all designed to fit into your basic 1K version of the **SINCLAIR ZX-80.** In programs which go far beyond anything that has been published the authors show the unique capabilities of the Sinclair ZX-80. The ZX-80 is more powerful than you ever thought!

BLACK JACK — Actually contains a full pack of cards, shuffles them, keeps track of the dealer and player card totals, and the money bet, all within 1K.

MEMORY LEFT — An incredible routine, especially useful as it enables you to know exactly how much memory is left, even during the running of a program. This also illustrates USR routines.

DR. ZX-80 — A conversational program with the computer as analyst which uses an ingenious method of storage.

GOMOKU — The computer challenges you to this complex Japanese game. Incredibly this program including display of the 7 x 7 board fits into 1K; it only does so because it uses the display as memory!

'This book is excellent!' – Clive Sinclair

Other programs included are HORSE RACE, LUNAR LANDER (with moving spaceship), NOUGHTS & CROSSES, NIM, SIMPLE SIMON, HANGMAN, LIFE, MASTERMIND, PINCH and 17 others.

As well as the programs, the book illustrates programming techniques you can use in your own programs — space comprehension, PEEKs and POKEs, USRs and so on. Offering program value of well over $40, the book is available for $12.50.

COMPSHOP AUSTRALIA

Compshop Australia
Office 4, 75 Palmerston Crescent
South Melbourne, Victoria.

I have a . . . . microcomputer.

☐ Please send me your software catalogue. I enclose a stamped self-addressed envelope.

☐ Please send me . . . .

I enclose a cheque/postal order for $ . . . .
(plus $1 for post and packing)

Signature: . . . .

Name: . . . .

Address: . . . .

. . . . Postcode: . . . .

APC-80

Many improvements available for the TRS-80 require some form of hardware modification but perhaps the most interesting require only a little delving into the heart of the TRS-80, its ROM, and some rearrangement of the way it conducts itself. In this first instalment of APC-80 Ian Davies establishes the new TRS-80 behaviour pattern and introduces five new commands to the Basic repertoire.

You may have wondered about that dark expanse of memory from 12288 to 17128 usually referred to as "reserved RAM". You may even have experienced the terror of a program over-writing part of it and fleeting off into a world of its own, interrupted only by a power-down. APC now has a peek at reserved RAM and how it can be used to turn your TRS-80 into an APC-80.

Much of the information in this series relates to the assembly language source listings. It is not necessary to have an understanding of assembly language, or how APC-80 uses reserved RAM, to be able to use the programs presented here. Detailed descriptions of the assembly language programs are included for those with an understanding of assembly language and as a guide to potential contributors.

Vectors

As a TRS-80 executes a Basic program, it branches out from ROM between every instruction to check the keyboard status. Therefore, there is actually more than one process being executed concurrently. This is only one example; there are many more – all vital to the machine staying in control of itself.

Most of these processes involve some form of vectoring, a concept used by the ROM's authors to make it possible to effectively grab hold of the computers attention as it emerges from ROM to begin a particular house-keeping duty such as scanning the keyboard. The TRS-80 would be a fairly inflexible beast if no provision were made for such program interruption. For example, should you want to service the keyboard in a special way that hadn't occurred to the ROM's authors, then you would have to implement a USR call to a machine language routine. While this solution would work, it's a "sledge-hammer" technique. It would require modification of your existing software to include the new input. Even then, the system functions (break, pause and command mode) would still use the normal input routine.

There are two forms of vectoring available. The first, used by ROM before calling a routine, involves looking at three bytes in reserved RAM. These usually contain a return instruction so the interpreter simply continues execution from where it left off. If, however, a jump instruction is encountered in these bytes, another routine has been specified and will be performed before a return to normal program execution is made.

The second type of vectoring is more direct. The interpreter loads a two byte value from the reserved RAM vector and executes the routine specified by that value.

The first method is used when the ROM routine must be executed, even if some other routine is to be used (the 'AND' vector). The other method is used when the ROM routine may be completely replaced by another (the 'OR' vector). For our purposes, the two methods can be considered functionally identical.

When the TRS-80 initializes, it moves values from ROM into reserved RAM thereby setting up the vectors for normal operation. Initialization occurs after a power-up and is the only way of performing a "true" reset (after all, you don't loose your Basic programs when you press the reset button. The pointers for your code are retained in reserved RAM).

Microsoft, the author of the Level II ROM, had enough foresight to provide vectors for most useful routines and checkpoints. There is no doubt this was done for their own use in the introduction of Disk Basic. But there is nothing to prevent mere mortals from using them either.

Auto repeat

The first routine in APC-80 adds a flashing block cursor and an auto repeat function to the TRS-80. As is standard on terminals, the auto repeat takes effect only when a key is held down for longer than about half a second after which that character is repeated ten times a second.

Once you have tried it, you can adjust it to suit your mood or application by POKEing a value into (for a 16k machine):

32240 for the repeat latency
32277 for the repeat frequency
32305 for the cursor flash rate.

Larger values increase the time factor in each case.

New commands

This is the part we are quite excited about as it has applications beyond the realm of normal protected memory routines. It introduces new commands to the TRS-80 in a very elegant and efficient manner.

Five new commands have been added. In future issues the list will be greatly expanded though you will not have to retype the routine. We'll be presenting a list of changes required to upgrade the superceded version.

The APC-80 Version 1 extended commands are:

CHAR 64 – will convert the screen back to sixty-four character mode without executing a CLS. This command's main use will be in graphical games though it can also be used to produce double spacing (see examples below).

CHAR 32 – has the same function as PRINT CHR$(23) but is more convenient to use.

BEEP <exp 1> , <exp 2> – where <exp > is a numeric expression of any sort, between O and 255 (it undergoes truncation if it is non-integral). The command outputs a square wave through the cassette Aux plug. Either connect a speaker to the earphone jack of the cassette recorder (and place the unit in record mode) or plug the Aux line into an amplifier. The generated frequency corresponds to the value of <exp 1> and duration to that of <exp 2>.

BUZZ < exp> – is similar to BEEP except that it uses the cassette relay to make a small buzzing sound. <exp> is the duration of the buzz and must have a value between O and 255. I must warn you that the cassette relays are not renowned for their long life and so prolonged use is not recommended.

DIE – is equivalent to typing SYSTEM and answering the prompt with a /O, but can be used within a program.

These commands can be used in programs or in the command mode. They are fully syntax checked and integrate smoothly with Level II and Disk Basic.

Getting started

If you own an editor assembler, type in the source code shown in Fig. 1. The ORG value is set for a 16k machine so other values will have to be inserted if you own a computer of different memory size (the ORG value should be 1k below the top of memory).

The Basic version is designed for a 16k TRS-80 and loads the machine code from DATA statements into protected high memory. To use APC-80 just type it in, or load it from cassette after setting the memory size to 31767 after power-up, and RUN it. The Basic program can then be cleared by typing NEW and APC-80 engaged by typing

SYSTEM and answering the prompt with a slash (/).

Future directions

We want to encourage your participation in this project and so we'd be pleased to hear from anyone with a routine to be added to the system or even just an idea but not a clue about how to implement it.

Just a few of the possible future routines that have already occurred to us are:

WAIT <exp >
DRAW X1, Y1, X2, Y2
ON BREAK GOTO
RENUMBER <start>,<increment>
MOVE <line number>, <line number>
FIND <string exp>
non destructive cursor
full cursor movement

If you're interested in contributing to APC-80, write to Ian Davies, APC, P.O. Box 115, Carlton, 3053.

How it works

Auto Repeat/Flashing Cursor

As can be seen in the assembly source listing (Fig. 1), the first task is to set up the relevant vector. This can be done directly with an editor assembler, but I encourage authors to employ the method shown as this will ensure compatibility with people using the Basic version.

The modified vector is normally used in the ROM keyboard scan routine. This routine is called between three and five hundred times a second when the machine is idle (waiting for input) and once between each Basic program statement execution.

The new values given to the vector direct the interpreter to a routine called REPEAT in high memory which saves the HL register pair and calls FLASH.

FLASH maintains a counter which is decremented once every call. If the counter is greater than zero, FLASH calls the ROM keyboard scan (to which the vector normally points) and then returns to REPEAT; if the counter is equal to zero, it is reset, the cursor toggled, ROM keyboard scan routine called and control returned to REPEAT.

REPEAT analyses the keyboard scan. If a key has been freshly pressed then the repeat counter is reset to pause 1, the value of the key stored and control passed back to ROM. If no keys have been freshly pushed, the routine checks whether any keys are down and if not returns to ROM. If any key has been pressed the repeat counter is decremented by one and control either returned to ROM (if the counter is greater than zero) or the value of the key saved and the counter reset to pause 2 (if the counter equals zero) before returning. This simulates a freshly pushed key. The routine also resets the FLASH counter to ensure it behaves correctly when auto-repeating.

New Commands

When Level II Basic encounters an error it immediately checks an 'OR' vector which Disk Basic resets to divert the interpreter to its own error diagnostic (producing, for example, SYNTAX ERROR instead of ?SN ERROR). If Disk Basic is not resident, the interpreter vectors back to Level II's diagnostic.

APC-80 sets the vector to point to the routine TRAPPR which instigates a return to the Basic error routine if the encountered error is not syntactical. A table search is then executed to determine if the error resulted from an APC-80 command. If so, program execution is directed to the appropriate routine which may reside in ROM, RAM or a mixture of both.

After performing the routine, the remaining portion of the current Basic program line is checked and errors indicated by a ?UE message. If no errors are encountered, the relevant pointers are adjusted and a smooth re-entry is made into Basic.

One complication does arise while using this vector. The ON ERROR command is consulted before the vector. Therefore, if the ON ERROR command is activated, when the interpreter encounters an APC-80 command and calls the error routine, the ON ERROR command will direct the interpreter to the required Basic line number. A solution is effected by the routine RESETR, which ensures the APC-80 table search is executed before the ON ERROR routine.

The advantages of trapping normal Basic interpreter function by resetting vectors is that all the facilities of Basic are retained (except for the full length diagnostic messages in Disk Basic) and extra commands appear as though they are part of Basic instead of requiring prefixes such as CMD "BEEP".

Fig. 1

```
7DDE                     ORG     3222 2

0300          PAUSE1     EQU     0300 H ; TIME UNTIL REPEAT EFFECT
0160          PAUSE2     EQU     0160 H  ; TIME BETWEEN REPEATS
3801          KEYBD      EQU     3801 H  ; ADDRESS OF KEYBOARD
06CC          BASIC      EQU     06CC H  ; ENTRY POINT TO BASIC
03E3          INPUT      EQU     0995 D  ; NORMAL INPUT VECTOR
0300          FTIME      EQU     0300 H  ; TIME BETWEEN FLASHES
4020          CURPOS     EQU     4020 H  ; LOCATION CONTAINING
                                           CURSOR POSITION
4016          INVECT     EQU     1640 6  ; LOCATION OF INPUT VECTOR

              ; PRELUDE CODE: KEYB OARD  VECTOR
              ;               SET- UP. EXECUTED
              ;               ONCE  ONLY.

7DDE 21EA7D   ENTRY      LD      HL,R EPEAT
7DE1 221640              LD      (INV ECT),HL
7DE4 310006              LD      SP,0 600H
7DE7 C3CC06              JP      BASIC

              ;   KEY  REPEAT  ROUTINE
              ;   ***************** *****

7DEA E5       REPEAT     PUSH    HL
7DEB CD1F7E              CALL    FLASH ; (AND INPUT ROUTINE)
7DEE 210003              LD      HL,PAUSE1
7DF1 A7                  AND     A
7DF2 2805                JR      Z,NOTNEW ; IF NOT FRESHLY PUSHED
7DF4 32307E              LD      (CHAR),A ; OTHERWISE REMEMBER IT
7DF7 1821                JR      REXIT ; AND GET OUT
7DF9 210138   NOTNEW     LD      HL,K EYBD
7DFC B6       KLOOP      OR      (HL); OK, ITS NOT FRESHLY PUSHED
7DFD CB25                SLA     L ; BUT LETS SEE IF ANY KEYS
7DFF F2FC7D              JP      P,KLOOP ; ARE DOWN ANYWAY....
7E02 A7                  AND     A
7E03 2815                JR      Z,REXIT    ; IF NOT, GIVE UP
7E05 2A3B7E              LD      HL,(RCOUNT)
7E08 2B                  DEC     HL ; OTHERWISE DECREMENT REPEAT COUNTER
7E09 7C                  LD      A,H
7E0A A7                  AND     A
7E0B 3E00                LD      A,0
7E0D 200B                JR      NZ,R EXIT   ; IF NOT ZERO, EXIT
7E0F 3E03                LD      A,03
7E11 32397E              LD      (FCOUNT),A ; OTHERWISE RESET THE COUNTER
7E14 216001              LD      HL,PAUSE2
7E17 3A3D7E              LD      A,(CHAR); SET A AS IF THE KEY IS FRESHLY
                                 PUSHED
7E1A 223B7E   REXIT      LD      (RCOUNT),HL
7E1D E1                  POP     HL
7E1E C9                  RET

              ;     FLASHER   ROUTINE
              ;     *************** ****

7E1F E5       FLASH      PUSH    HL
7E20 2A397E              LD      HL,(FCOUNT)
7E23 2B                  DEC     HL ; DECREMENT FLASH COUNTER
7E24 7C                  LD      A,H
7E25 A7                  AND     A
7E26 200A                JR      NZ,FSKIP  ; IF NOT ZERO, LEAVE IT ALONE
7E28 2A2040              LD      HL,(CURPOS)
7E2B 7E                  LD      A,(HL)
7E2C EE00                XOR     0D0H ; OTHERWISE TOGGLE IT
                         LD      (HL) 7E2E 77 ,A
7E2F 210003              LD      HL,F TIME    ; AND RESET COUNTER
7E32 22397E   FSKIP      LD      (FCO UNT),HL
7E35 E1                  POP     HL
7E36 C3E303              JP      INPUT        ; AND VECTOR OFF.

7E39 0003     FCOUNT     DEFH    FTIM E
7E3B 0003     RCOUNT     DEFH    PAUS E1
7E3D 00       CHAR       DEFB    0H
7DDE                     END     ENTR Y
7F05                     ORG     32517

7F05 3EC3     INITS      LD      A,0C3H ; SET UP VECTORS
7F07 32A641              LD      (41A 6H),A
7F0A 32C441              LD      (41C 4H),A
7F0D 21507F              LD      HL,T RAPPR
7F10 22A741              LD      (41A 7H),HL
7F13 213A7F              LD      HL,R ESETR
7F16 22C541              LD      (41C 5H),HL
7F19 21257F              LD      HL,M ESSGE  ; GIVE MESSAGE
7F1C CDA728              CALL    28A7 H
7F1F 310006              LD      SP,0 600H
7F22 C3CC06              JP      06CC ; ENTER COMMAND MODE

7F25 1C1F     MESSGE     DEFW    1F1C H   ; HOME, CLR
7F27 41                  DEFM    APC -80  VERSION 1'
7F38 1A00                DEFW    001A H   ; LF,END

7F3A E5       RESETR     PUSH    HL
7F3B 2A2040              LD      HL,(4020H)
7F3E 22FC7F              LD      (CURSA),HL
7F41 2AF140              LD      HL,(40F1H)
7F44 7C                  LD      A,H
7F45 2F                  CPL
7F46 A5                  AND     L
7F47 280C                JR      Z,NO MOVE
7F49 2AF040              LD      HL,(40F0H)
7F4C 22F97F              LD      (ERR VEC),HL
7F4F 210000              LD      HL,0 000H
7F52 22F040              LD      (40F 0H),HL
7F55 3AF240   NOMOVE     LD      A,(4 0F2H)
7F58 32FE7F              LD      (FLA G2),A
7F5B E1                  POP     HL
7F5C C9                  RET
```

```
                00450
                00460
                00470 ;   THIS IS THE DECODER THAT JUMPS IN WHEN AN
                00480 ;   ERROR OCCURS. IT OPERATES TRANSPARENTLY TO
                00490 ;   THE INTERPRETER.
                00500
7F5D F5         00510 TRAPPR  PUSH    AF
7F5E E5         00520         PUSH    HL
7F5F 3A9A40     00530         LD      A,(409AH)  ; GET ERROR CODE
7F62 D602       00540         SUB     02H ; MUST BE ?SN ERROR
7F64 67         00550         LD      H,A
7F65 3AFB7F     00560         LD      A,(FLAG) ; CHECK FOR RECURSIVE ENTRY
7F68 B4         00570         OR      H
7F69 2067       00580         JR      NZ,QUIT
7F6B D5         00590         PUSH    DE ; MAKE SOME ROOM
7F6C C5         00600         PUSH    BC
7F6D 11587E     00610         LD      DE,TABLE   ; PTR TO TABLE
7F70 2AEE40     00620         LD      HL,(40EEH) ; PTR TO CODE
7F73 7E         00630         LD      A,(HL)
7F74 A7         00640         AND     A
7F75 200C       00650         JR      NZ,IMMED    ; IF IMMED MODE
7F77 ED4BFC7F   00660         LD      BC,(CURSA)
7F7B ED432040   00670         LD      (4020H),BC
7F7F 010400     00680         LD      BC,0004
7F82 09         00690         ADD     HL,BC
7F83 22F77F     00700 IMMED   LD      (START),HL ; PLACE TO SCAN FROM
                00710
7F86 2AF77F     00720 WORDLP  LD      HL,(START) ; GOTO START OF LINE
7F89 1A         00730         LD      A,(DE)      ; GET WORD LENGTH
7F8A 47         00740         LD      B,A        ; COUNT IN REG. B
7F8B A7         00750         AND     A   ; SET FLAGS
7F8C 2842       00760         JR      Z,WINDUP    ; IF END OF TABLE
7F8E D7         00770 CHARLP  RST     10H ; SKIP, GET & SET
7F8F 283F       00780         JR      Z,WINDUP    ; IF END OF LINE
7F91 EB         00790         EX      DE,HL       ; SWAP
7F92 23         00800         INC     HL
7F93 BE         00810         CP      (HL) ; ACTUALLY DE
7F94 EB         00820         EX      DE,HL       ; SWAP THEM BACK
7F95 201A       00830         JR      NZ,NEXTWD   ; IF <> GOTO NEXT WORD
7F97 10F5       00840         DJNZ    CHAR LP     ; CONTINUE IF STILL =
                00850
                00860 ;  DROPS THROUGH HERE IF THE TABLE ENTRY IS A
                00870 ;  DIRECT MATCH WITH THE FIRST "SYMBOL" OF
                00880 ;  THE LINE IN ERROR
                00890
7F99 3EFF       00900         LD      A,255
7F9B 32FB7F     00910         LD      (FLAG),A
7F9E 32EA40     00920         LD      (40EAH),A  ; TO END NICELY
7FA1 32EB40     00930         LD      (40EBH),A
7FA4 13         00940         INC     DE ; POINTS TO VECTOR
7FA5 01B87F     00950         LD      BC,RETURN
7FA8 C5         00960         PUSH    BC ; RETURN ADDRESS ON STACK
7FA9 E5         00970         PUSH    HL
7FAA C1         00980         POP     BC ; MOVE SOURCE PTR TO BC
7FAB EB         00990         EX      DE,HL ; HL POINTS TO VECTOR
7FAC 5E         01000         LD      E,(HL)
7FAD 23         01010         INC     HL
7FAE 56         01020         LD      D,(HL)
7FAF EB         01030         EX      DE,HL       ; HL CONTAINS VECTOR
7FB0 E9         01040         JP      (HL) ; "CALL" ROUTINE
                01050
                01060
7FB1 13         01070 NEXTWD  INC     DE ; SKIP REST OF WORD
7FB2 13         01080         INC     DE ; IN TABLE AND ITS
7FB3 13         01090 SKIPCH  INC     DE ; VECTORS.  POINT
7FB4 10FD       01100         DJNZ    SKIPCH     ; AT NEXT LENGTH.
7FB6 18CE       01110         JR      WORDLP     ; DO IT FOR NEXT WORD
                01120
                01130
7FB8 C5         01140 RETURN  PUSH    BC ; GET IT BACK FOR BASIC
7FB9 E1         01150         POP     HL
7FBA C1         01160         POP     BC
7FBB D1         01170         POP     DE
7FBC AF         01180         XOR     A
7FBD 32FB7F     01190         LD      (FLAG),A
7FC0 3AFE7F     01191         LD      A,(FLAG2)
7FC3 32F240     01192         LD      (40F2H),A
7FC6 F1         01200         POP     AF ; CLEAR OFF STACK
7FC7 F1         01210         POP     AF
7FC8 F1         01220         POP     AF
7FC9 D7         01230         RST     10H ; CHECK FOR ERRORS AFTER IT
7FCA CA1E1D     01240         JP      Z,1D1EH     ; IF OK
7FCD C30320     01250         JP      2003H       ; ELSE ?EU ERROR
                01260
7FD0 C1         01270 WINDUP  POP     BC
7FD1 D1         01280         POP     DE
7FD2 E1         01290 QUIT    POP     HL
7FD3 3AFE7F     01300         LD      A,(FLAG2) ; CHECK ON ERROR STATUS
7FD6 2F         01310         CPL
7FD7 57         01320         LD      D,A
7FD8 3AFA7F     01330         LD      A,(ERRVEC+1) ; AND ON ERROR POINTER
7FDB A2         01340         AND     D
7FDC 2813       01350         JR      Z,BOMB          ; FOR ERROR MESSAGE
7FDE AF         01360         XOR     A
7FDF 32FB7F     01370         LD      (FLAG),A
7FE2 F1         01400         POP     AF
7FE3 3A9A40     01410         LD      A,(409AH)     ; GET ERROR CODE
7FE6 5F         01420         LD      E,A
7FE7 E1         01430         POP     HL ; CLEAR STACK
7FE8 2AF97F     01440         LD      HL,(ERRVEC)
7FEB 22F040     01450         LD      (40F0H),HL
7FEE C3A219     01460         JP      19A2      ; EXECUTE "ON ERROR" CODE
7FF1 AF         01470 BOMB    XOR     A
7FF2 32FB7F     01480         LD      (FLAG),A
7FF5 F1         01490         POP     AF
7FF6 C9         01500         RET
                01510
                01520
                01530
7FF7 0000       01540 START   DEFW    0
7FF9 0000       01550 ERRVEC  DEFW    0
7FFB 00         01560 FLAG    DEFB    0
7FFC 0000       01570 CURSA   DEFW    0
7FFE 00         01580 FLAG2   DEFB    0
                01590
                01600
                01610
                01620
                01630 ;   THIS IS THE SYMBOL TABLE FOR USE BY THE DECODER
                01640 ;   IT CONTAINS 3 ELEMENTS FOR EACH SYMBOL:-
                01650 ;              THE LENGTH IN CHARACTERS
                01660 ;              THE NAME
                01670 ;              THE VECTOR FOR ITS ROUTINE
                01680 ;
                01690 ;   A LENGTH OF ZERO ENDS THE TABLE. THE TABLE SHOULD
                01700 ;   BE PLACED WITH PLENTY OF ROOM FOR EXPANSION.
                01710
                01720
7E58            01730         ORG     32347
                01740
7E58 06         01750 TABLE   DEFB    6
7E5C 43         01760         DEFM    'CHAR32'
7E62 F604       01770         DEFW    04F6 H
                01780
7E64 06         01790         DEFB    6
7E65 43         01800         DEFM    'CHAR64'
7E6B C304       01810         DEFW    04C3 H
                01820
7E6D 03         01830         DEFB    3
7E6E 44         01840         DEFM    'DIE'
7E71 0000       01850         DEFW    0000
                01860
7E73 04         01870         DEFB    4
7E74 42         01880         DEFM    'BEEP'
7E78 E07E       01890         DEFW    BEEP
                01900
7E7A 04         01910         DEFB    4
7E7B 42         01920         DEFM    'BUZZ'
7E7F 3E7E       01930         DEFW    BUZZ
                01940
7E81 00         01950         DEFB    0 ; END OF TABLE
                01960
                01970
                01980
                01990 ;   THIS IS THE BEEP ROUTINE, IT IS PLACED
                02000 ;   DIRECTLY BELOW THE INITIALISATION ROUTINE.
                02010
                02020
7EE0            02030         ORG     32480
                02040
7EE0 C5         02050 BEEP    PUSH    BC
7EE1 E1         02060         POP     HL
7EE2 23         02070         INC     HL ; SET UP SOURCE PTR IN HL
7EE3 CD1C2B     02080         CALL    2B1CH      ; READ FIRST PARAMETER
7EE6 1C         02090         INC     E
7EE7 D5         02100         PUSH    DE ; SAVE VALUE ON STACK
7EE8 3E2C       02110         LD      A,2CH
7EEA BE         02120         CP      (HL)
7EEB C24A1E     02130         JP      NZ,1E4AH    ; CHECK FOR COMMA
7EEE D7         02140         RST     10H
7EEF CD1C2B     02150         CALL    2B1CH      ; GET SECOND PARAMETER
                02160
                02170 ;  START OF SOUND GENERATOR
                02180
7EF2 3C         02190         INC     A
7EF3 C1         02200         POP     BC ; FREQ IN C
7EF4 57         02210         LD      D,A ; DURATION IN D
7EF5 3E01       02220         LD      A,01 ; SET UP BIT PATTERN
7EF7 EE03       02230 LOOP    XOR     03   ; TOGGLE BIT PATTERN
7EF9 D3FF       02240         OUT     (0FFH),A   ; AND OUTPUT IT
7EFB 41         02250         LD      B,C ; MOVE FREQ INTO B
7EFC 10FE       02260 FREQ    DJNZ    FREQ ; AND DELAY
7EFE 15         02270         DEC     D
7EFF 20F6       02280         JR      NZ,LOOP     ; LOOP FOR DURATION
7F01 2B         02290         DEC     HL ; ADJUST SOURCE PTR
7F02 E5         02300         PUSH    HL
7F03 C1         02310         POP     BC ; AND LEAVE BACK IN BC
7F04 C9         02320         RET
                02330
                02340
                02350 ;  BUZZ ROUTINE.
                02360 ;  NO FREQUENCY PARAMETER, JUST DURATION
                02370
7E3E            02380         ORG     32318
                02390
7E3E C5         02400 BUZZ    PUSH    BC
7E3F E1         02410         POP     HL
7E40 23         02420         INC     HL
7E41 CD1C2B     02430         CALL    2B1CH ; GET DURATION INTO E
7E44 1C         02440         INC     E
7E45 3A3D40     02450         LD      A,(403DH)  ; GET SCREEN MODE
                02460
7E48 EE04       02470 ZLOOP   XOR     4 ; TOGGLE OUTPUT VALUE
7E4A D3FF       02480         OUT     (0FFH),A    ; GIVE IT TO RELAY
7E4C 067F       02490         LD      B,7FH       ; FREQUENCY
7E4E 10FE       02500 ZFREQ   DJNZ    ZFREQ       ; WAIT HERE
7E50 1D         02510         DEC     E
7E51 20F5       02520         JR      NZ,ZLOOP    ; LOOP FOR THE DURATION
7E53 E6FB       02530         AND     0FBH
7E55 D3FF       02540         OUT     (0FFH),A    ; MAKE SURE RELAY IS OFF
7E57 2B         02550         DEC     HL ; ADJUST SOURCE PTR
7E58 E5         02560         PUSH    HL
7E59 C1         02570         POP     BC ; PUT IT BACK INTO BC
7E5A C9         02580         RET
                02590
                02600
7F05            02610         END     INITS
  BASIC   06CC 00007   00022
  CHAR    7E3D 00076   00033 00050
  CURPOS  4020 00010   00065
  ENTRY   7DDE 00019   00077
  FCOUNT  7E39 00074   00048 00060 00070
  FLASH   7E1F 00059   00029
  FSKIP   7E32 00070   00064
  FTIME   0300 00009   00069 00074
  INPUT   03E3 00008   00072
  INVECT  4016 00011   00020
  KEYBD   3801 00006   00035
  KLOOP   7DFC 00036   00038
  NOTNEW  7DF9 00035   00032
  PAUSE1  0300 00004   00030 00075
  PAUSE2  0160 00005   00043
  RCOUNT  7E3B 00075   00041 00051
  REPEAT  7DEA 00028   00019
  REXIT   7E1A 00051   00034 00040 00046
  BEEP    7EE0 02050   01890
  BOMB    7FF1 01470   01350
  BUZZ    7E3E 02400   01930
  CHARLP  7F8E 00770   00840
  CURSA   7FFC 01570   00300 00660
  ERRVEC  7FF9 01550   00370 01330 01440
  FLAG    7FFB 01560   00560 00910 01190 01
  370 01480
  FLAG2   7FFE 01580   00410 01191 01300
  FREQ    7EFC 02260   02260
  IMMED   7F83 00700   00650
  INITS   7F05 00120   02610
  LOOP    7EF7 02230   02280
  MESSGE  7F25 00240   00190
  NEXTWD  7FB1 01070   00830
  NOMOVE  7F55 00400   00350
  QUIT    7FD2 01290   00580
  RESETR  7F3A 00280   00170
  RETURN  7FB8 01140   00950
  SKIPCH  7FB3 01090   01100
  START   7FF7 01540   00700 00720
  TABLE   7E58 01750   00610
  TRAPPR  7F5D 00510   00150
  WINDUP  7FD0 01270   00760 00780
  WORDLP  7F86 00720   01110
  ZFREQ   7E4E 02500   02500
  ZLOOP   7E48 02470   02520
```

Fig. 2

```
10 CLS : POKE 16553,255
20 PRINT "LOADING    A. P. C. - 8 0"
30 READ H$
40 C%=C%+1 : PRINT @ 56, C%
50 IF LEN(H$)>2 THEN 110
60 H%=ASC(LEFT$(H$,1))-48
70 L%=ASC(RIGHT$(H$,1))-48
80 IF H%>9 THEN H%=H%-7
90 IF L%>9 THEN L%=L%-7
100 POKE B%+O%,H%*16+L%: O%=O%+1:GOTO 30
110 IF H$="ORG" THENREAD B%:O%=0:GOTO 30
120 IF H$<>"END" THEN STOP
130 PRINT
140 PRINT "LOAD COMPLETE"
150 POKE 16607,5
160 POKE 16608,127
170 END
180 REM  FLASHING CURSOR & AUTO REPEAT
190 DATA ORG,32222
200 DATA 21, EA, 7D, 22, 16, 40, 31, 00
210 DATA 06, C3, CC, 06, E5, CD, 1F, 7E
220 DATA 21, 00, 03, A7, 28, 05, 32, 3D
230 DATA 7E, 18, 21, 21, 01, 38, B6, CB
240 DATA 25, F2, FC, 7D, A7, 28, 15, 2A
250 DATA 3B, 7E, 2B, 7C, A7, 3E, 00, 20
260 DATA 0B, 3E, 03, 32, 39, 7E, 21, 60
270 DATA 01, 3A, 3D, 7E, 22, 3B, 7E, E1
280 DATA C9, E5, 2A, 39, 7E, 2B, 7C, A7
290 DATA 20, 0A, 2A, 20, 40, 7E, EE, D0
300 DATA 77, 21, 00, 03, 22, 39, 7E, E1
310 DATA C3, E3, 03, 00, 03, 00, 03, 00
320 REM  APC-80 DECODER ROUTINE
330 DATA ORG, 32517
340 DATA 3E, C3, 32, A6, 41, 32, C4, 41
350 DATA 21, 5D, 7F, 22, A7, 41, 21, 3A
360 DATA 7F, 22, C5, 41, 21, 25, 7F, CD
370 DATA A7, 28, 31, 00, 06, C3, DE, 7D
380 DATA 1C, 1F, 41, 50, 43, 2D, 38, 30
390 DATA 20, 20, 56, 45, 52, 53, 49, 4F
400 DATA 4E, 20, 31, 1A, 00
410 '
420 DATA E5, 2A, 20, 40, 22, FC, 7F, 2A
430 DATA F1, 40, 7C, 2F, A5, 28, 0C, 2A
440 DATA F0, 40, 22, F9, 7F, 21, 00, 00
450 DATA 22, F0, 40, 3A, F2, 40, 32, FE
460 DATA 7F, E1, C9, F5, E5, 3A, 9A, 40
470 DATA D6, 02, 67, 3A, FB, 7F, B4, 20
480 DATA 67, D5, C5, 11, 58, 7E, 2A, EE
490 DATA 40, 7E, A7, 20, 0C, ED, 4B, FC
500 DATA 7F, ED, 43, 20, 40, 01, 04, 00
510 DATA 09, 22, F7, 7F, 2A, F7, 7F, 1A
520 DATA 47, A7, 28
530 DATA 42, D7, 28, 3F, EB, 23, BE, EB
540 DATA 20, 1A, 10, F5, 3E, FF, 32, FB
550 DATA 7F, 32, EA, 40, 32, EB, 40, 13
560 DATA 01, B8, 7F, C5, E5, C1, EB, 5E
570 DATA 23, 56, EB, E9, 13, 13, 13, 10
580 DATA FD, 18, CE, C5, E1, C1, D1, AF
590 DATA 32, FB, 7F, 3A, FE, 7F, 32, F2
600 DATA 40, F1, F1, F1, D7, CA, 1E, 1D
610 DATA C3, 03, 20, C1, D1, E1, 3A, FE
620 DATA 7F, 2F, 57, 3A, FA, 7F, A2, 28
630 DATA 13, AF, 32, FB, 7F, F1, 3A, 9A
640 DATA 40, 5F, E1, 2A, F9, 7F, 22, F0
650 DATA 40, C3, A2, 19, AF, 32, FB, 7F
660 DATA F1, C9, 00, 00, 00, 00, 00, 00
670 DATA 00, 00
680 REM  SYMBOL AND VECTOR TABLE
690 DATA ORG, 32347
700 DATA 06, 43, 48, 41, 52, 33, 32, F6
710 DATA 04, 06, 43, 48, 41, 52, 36, 34
720 DATA C3, 04, 03, 44, 49, 45, 00, 00
730 DATA 04, 42, 45, 45, 50, E0, 7E, 04
740 DATA 42, 55, 5A, 5A, 3E, 7E, 00
750 REM  BEEP ROUTINE
760 DATA ORG, 32480
770 DATA C5, E1, 23, CD, 1C, 2B, 1C, D5
780 DATA 3E, 2C, BE, C2, 4A, 1E, D7, CD
790 DATA 1C, 2B, 3C, C1, 57, 3E, 01, EE
800 DATA 03, D3, FF, 41, 10, FE, 15, 20
810 DATA F6, 2B, E5, C1, C9
820 REM  BUZZ ROUTINE
830 DATA ORG, 32318
840 DATA C5, E1, 23, CD, 1C, 2B, 1C, 3A
850 DATA 3D, 40, EE, 04, D3, FF, 06, 7F
860 DATA 10, FE, 1D, 20, F5, E6, FB, D3
870 DATA FF, 2B, E5, C1, C9
880 DATA END
```

```
10 '  EXAMPLE PROGRAM
20 'MISSILE SHOT 1
30 FORI%=30TO1 STEP-2:BEEPI%,12:NEXT
40 STOP
50 'MISSILE SHOT 2
60 FORI%=1TO100STEP3:BEEPI%,30:NEXT
70 STOP
80 'TECKO SOUND
90 FORI%=1TO100:BEEP RND(120),RND(100):NEXT
100 STOP
110 'ALARM
120 FORI%=1TO20:FORK%=90TO50STEP-1:BEEPK
%,10:NEXT K%,I%
130 STOP
140 INPUT "WHAT'S YOUR NAME ";N$
150 IF N$="JOHN" THEN: BUZZ 255: ELSE IF N$="PETER" THEN: DIE
160 STOP
170 'DOUBLE SPACING
180 CHAR32:PRINT"APC-80 VERSION 1":CHAR64
190 END
```

DEFOREST SOFTWARE

**TRS-80 is a Trademark of the Tandy Corporation.*

GAMES

TRS-80* (MODEL 1)

Sargon 2 – The ultimate Chess Game for any Microcomputer 16K Level 2. . . . . $37.50
Microchess – Level 1 or 2 TRS-80. Plays an entertaining game 3 levels fits in 4K. $24.00
Golf & Crossout – New improved version, good graphics. You can even design your own championship course. $10
Santa Paravia – up to 12 can play, become the king of a medevil city, levee taxes $10.00
Battleground – The stark reality of World War II comes alive in this program $12.50
Game Playing with Basic – 3 tapes 1, 2 & 3 based on the book of the same name great for learning . . . $12.50 ea.
Oil Tycoon – 2 players, explore, drill for oil, outprice the opposition, force him into bankruptcy. $10.00
Space Trek IV – population simulation, trade or wage war send missions to space, can you survive? $10.00
Checker King – not just an ordinary checker game but a fast expert machine language challenge. $27.00
Flight – Control your own aircraft, very realistic, good flying simulation and very challenging $10.00
Jet Fighter Pilot – Takes you as close to real combat fighting as possible. A brilliantly realistic simulation $19.95
Airmail Pilot – an entertaining game can you get the mail through in your 1927 Biplane, watch for lightening, windstorms, fuel shortages etc.. . . . . $10.00
Night Flight – a new exciting game from Instant Software $12.50
Flight Path – from Instant Software $12.50
Adventures – what more can be said! The most challenging simulation/game/adventure ever Nos. 0 to 10 . ea. $16.00
1 on Disc $19.00
2 on Disc $32.00
3 on Disc $39.00
Adventure 10 now available – "Savage Island" (Part One)
Backgammon/Keno – 2 great gambling games from instant software. $10.00
Mind Warp – A series of great programmes $12.50
I.Q. Test – See how smart you really are $12.50
Interactive Fiction – "Little Red Riding Hood" Computer adventures into the children's world of make believe $17.50
Omni Calculator – Disk – an invaluable air for teachers, doctors, surveyors, pharmacists etc. . . . $19.95
Dynamic Device Driver – programmable keyboard lock – l/c support – repeat key – forms control etc. $24.00

Olympic Decathlon by Microsoft Disk $29 00; Cassette $29.00;
Bandito – Your TRS-80 a slot machine with great sound effects $12.50
Star Trek – Acorn a space adventure on your TRS-80 . $12.50
Stock Market – Invest in safe stocks or high risk specs. will you take the risk $12.50
Original Adventure – requires 32K & disc from Microsoft. $35
Monolopy – Play that famous game against the TRS-80. . $10
Time Trek – with sound, another of the 'classics' for the TRS-80 real time excellent programme $21.00
Galactic Empire – a space game with a difference, many hours enjoyment with this one. $16.00
Cosmic Patrol (with sound) – WARNING! Players of this game should be prepared for a state of realism hitherto unavailable on the TRS-80 . . . . $19.95
Light Pen – games and instructional program includes game frogs (Light pen not included) $10.00
Space Battles – requires 32K 1 disc $15.00
Games 20 – only 75 cents per program on disk $15.00
Pinball – the greatest game that I have ever seen for the TRS-80, fast action movable flippers and great sound. Machine language, maintains high score Disk. $20
Tape. $16
Super Space Invaders with Sound – nothing like the normal invaders extra fast. Spray bullets like out of a hosepipe, not only does the enemy move sideways but they also advance . . . . .Disk $20
Tape $16
Space War – A 2 pleyer real time space battle; 5 modes, 2 speeds $12.50
Ultra Trek – A Star Trek theme in a challenging adult version $19.95
Lying Chimps with Sound – a very interesting program . . $10
Chessmate 80 – This programme forced a draw with Sargon 2 . . . . $24
Othello – An old favorite $12.50
Typing Teacher – $10
Your Cribbage & Checkers Partner $12.50
Quad – A tic-tac-toe strategy game with a difference . $12.50
Zossed in Space – machine language, with graphics . $17.50
Interactive Fiction – "Local Call for Death" you actually take part in history $22.95
Basic Program Assistant – list variables, lines with keywords etc. – "an invaluable aid to programming" $19.95
Disc Editor – list or modify any hyte on a disk . . $45.00
Mail File from Galactic – 600 names per disk. File may be split over disks $110.00

PET

TREK X with sound (for R&T). $10
Backgammon and Keno $10
Space Trek 2. . . . . $10
Code Breaker with sound $17.40
Some common basic programmes. $17.50
Dungeon of Death $10
Chimera $12.50
Santa Paravia $12.50

APPLE

Golf. $10
Bowling - Trilogy $10
Oil Tycoon. $12.50
Sahara Warriors $10
Space Wars. $10
Air Flight Simulation. $12.50
Santa Paravia $12.50
Math Fun – Change an Apple computer into a mathematics tutor and change boredom into enthusiasm (all programs in this package run in Applesoft basic except whole space which requires Integer Basic) $25
Finance and Investment – disk $22.50
Paddle Fun – disk $25.00
APPLE FUN comprising MIMIC, FLIGHT SIMULATION, COLOURMASTER, STARSHIP ATTACK and TRILOGY (Apple II or IIplus - 32K - 1 Drive - Mimic requires Applesoft in Rom) $25
Super Shooter – req. 24K integer – includes Space Invaders and Howitzer $12.50
Skybombers – 32K disk $22.50
Accounting Assistant – 16K $12.50

COMPUTER BOOKS

Basic Basic $11.00
Advanced Basic $11.00
Basic from the Ground up $11.00
Basic Work-book $7.50
Discovering Basic $8.50
Common Basic Programs $12.50
Sargon Chess $19.00
How to Build a computer controlled Robot $15.00
How to profit from your personal computer $11.00
An Introduction to Microprocessors 0 . . . . $11.75
" " " " 1 . . . . $12.50
Z80 Programming for Logic Design $12.50
Z80 Assembly Language Programming $13.60
Pay Roll & Cost Accounting* *(Software to match $99)* . . . $20.00
Account Payable & Receivable* *(Software to match $99)* . $20.00
General Ledger* *(Software to match $99)* $20.00
6502 Assembly Language Programming $13.60
The 6800 Microprocessor $11.00
Basic Microprocessors & the 6800 . . . . $16.00
Computer Mathematics $16.00
Consumers guide to personal computing & Microprocessors $11.00
Mini Computers, structure & programming $18.00
Fundamentals & Applications of Digitallogic circuits $13.00
Small Computer systems Handbook $11.50
The first book of Microcomputers $6.50
Computers in Society $9.50
Computers in Action $7.00
Standard Dictionary of Computers $25.00
Programming Programmable calculators $13.50
Cobol with style $9.50
Basic with style $8.00
Fortran with style $9.50
Pascal with style $8.50
Programming Proverbs $9.50
Fortran Fundamentals $6.50
Fortran 4 Programming $9.00
Microcomputer System Design $21.00
Microprocessor Data Manual $10.00
Microprocessor Basics $16.00
S-100 Handbook $16.00
Digital experiments $11.00
Digital Trouble Shooting $12.50
Telephone Accessories $7.50
More telephone accessories $7.50
Logical Design Using IC's $24.50
Statistical Pattern recognition $21.50
Fundamentals of Data Base Systems $21.00
400 Ideas for Design Vol 3 . . . . $18.50
Integrity & recovery in Computer Systems $12.50
Management of Information Systems $11.00
File Structure for On-Line Core Systems $18.50
Data Management for On-Line Core Systems $18.50
Digital Signal Analysis $27.00
Computer Security Risk Analysis $16.00
Character readers & Pattern recognition $17.50
Modern Electronics Security Systems $16.00
Printed Circuit Assembly $6.50
Basic Mathematics Vol 1 . . . . $8.50
Vol 2 . . . . $8.50
Mathematics for Electronics $9.50
Human Communication Handbook Vol 1 . . . . $12.00
Vol 2 . . . . $12.00
Computer Dynamics $13.00
Computer Aided Design Techniques $32.00
Electronic Game Projects $6.00
TRS-80 Disk and other Mysteries "Pennington" $22.50

★ *NEW!! Z80 soft card $365.00 (run Z80 programs on your APPLE).*

WE ARE NOW SOLE AUSTRALIAN AGENTS FOR INSTANT SOFTWARE

DUE TO POOR QUALITY FROM SOME MANUFACTURERS WE ARE ONLY STOCKING SCOTCH BRAND COMPUTER CASSETTES "MAGNIFICENT MEDIA"

DEFOREST SOFTWARE

The Floppy Doctor. Checks every sector of disk, motor speed head positioning, controller functions, status bits, etc. $25.00

New DOS 80 — The latest and greatest from Apparat, variable length fields to 4K, mix and match drives on 1 cable. 18 to 80 track. Security boot and much more . . . $149.50

New DOS Plus — A very powerful disk operating system, too many features to list. Send SAE for reviews and information . . . 40 track $110.00; 35 track $99.00; 77 track $120.00

New DOS — A scaled down version of the above . . . 40 track $55.00; 35 track $49.00

Editor Assembler Plus — Microsoft — Very powerful complete with de-bugger . . . $34.00

Basic Compiler — Create fast Z80 code of your basic programs $202.00

Level 3 Basic — get all the advantages of disk basic plus many extra features (disk or cassette) . . . $52.00

Assembly Language Package — Complete Marco Assembly by Microsoft . . . $102.00

Fortran — Microsoft — Fully compatable with TRS-DOS $102.00

Packer — Comes with 16K, 32K & 48K versions; perhaps the best program written in 1979; remove rems & spaces, renumber, move and more — Disk add $5.00. $30.00

Clone 2 — Copies system or data tapes, make backups of your valuable programs . . . $18.00

Print to L. Print — No Printer? Simply use this program to change all L. Prints to Prints or vice versa, other features. $10.00

System Savers — FLEXI and T DISK put your favourite M/L tape to disk or make back-ups of system tapes. $24.00

Tape based labelling disassembler — this program assigns labels (where appropriate) to the routines in a machine language program. The output is almost identical to hand assembled sources cose. Because of the use of labels, it is a simple matter to make changes without loosing track of jump/load addresses . . . $19.95

Disk-based labelling disassembler — as above but for disk systems . . . $24.95

Utility 1 & 2 — from Instant Software — 2 tapes

New DOS 80 — comes in a brown and gold blocked plastic cover with orange pages if your copy is different contact us or Apparat. It could be a forgery.

EDUCATION

SBT — Structure Basic Translator — write basic in a structured format . . . disk $23.00

Teacher — Create your own texts store on data file $12.50

French/Italian/German — Each on disk, very powerful language CAI program . . . each $23.00

Assembly Language Instruction Course — Audio cassettes, program cassette and reference book. Learn Assembly Language the easy way. This course guides you smoothly through the complexity of Assembly Language without difficulty . . . $70.00

HARDWARE

Speed up now to 100% faster. Make your machine the fastest on the block (new improved model, now 4 speed) $35.00

Coding Forms . . . 2 pads $8.50

Flow Chart Pads . . . 5 pads $37.50

Giant Video Display Pads . . . 5 pads $37.50

House brand 5¼ Diskettes . . . 10 for $40.00

Blank Computer Cassettes (Scotch) . . . 5 for $12.00; 10 for $19.50

** Due to poor quality from some manufacturers, we are only stocking Scotch brand cassettes "Magnificent Media".*

Data Separators . . . $29.95

Fine Line Graphics . . . $439.00

16K memory, 250ns . . . $57.00

Installation of above . . . $20.00

Installation of speed up mod. . . . $45.00

Level 2 Roms — "New Style" . . . $175.00

Installation . . . $25.00

Doubler — Double density for your machine . . . $250.00

OTHER PROGRAMS

Terminal features multi-protocol capability — auto repeat keys — true break key — soft keys, user programmable — forward or reverse scrolling — receive to disk file, printer or buffer. $88.00

The Electric Paintbrush — A graphics language, very powerful $21.00

Stimulating Simulations — from Personal Software . . . $21.00

General Mathematics . . . $12.50

CCA DMS — Data Base rep. 32K . . . Disk $95.00

MU MATH from MICROSOFT — a fantastic mathematic program . . . Disk $77.00

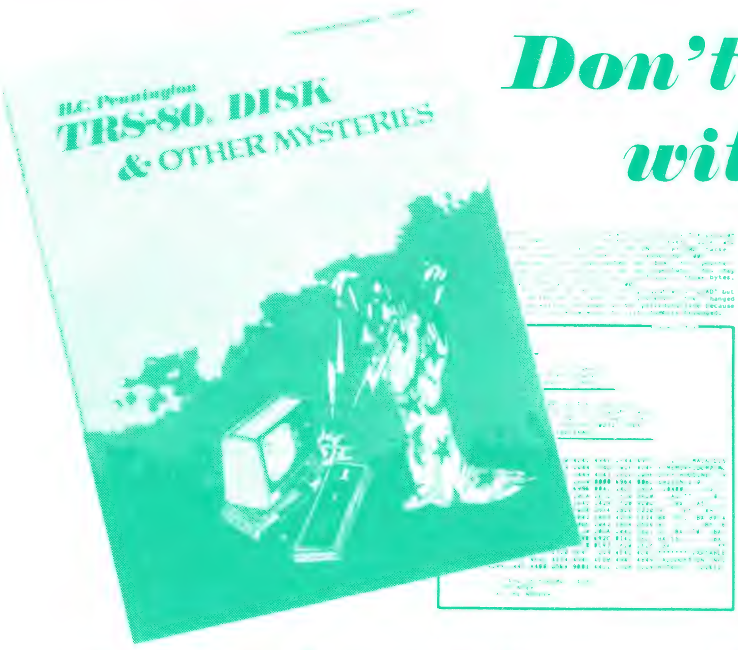
Azimuth Finder — Amateur Radio gives beam heading, call signs & distance for 100's of countries screen or printer $14.00

Ham Package — Instant Software . . . $10.00

Music Tutor (Write music on your TRS-80) . . . $12.00

Opera by Acorn — got to be heard to be believed . . . $16.00

Engineers — Beam Analysis Program . . . $25.00

Common basic programs (matches Osborne book) . . . $17.50

Don't boot-up without it!

The 'How To' book of TRS-80® data recovery

- 130 page, 8½ x 11 softcover
- Actual memory maps, samples
- Detailed recovery instructions
- Recover lost programs, DOS errors
- Fix parity, gat and hit errors
- Recover from unreadable directory
- Fix Electric Pencil file errors
- Recover lost or killed files

TRS-80 IS A REGISTERED TRADEMARK OF TANDY CORP

JUST $22.50

Get it and unveil the many disk mysteries

I'M A BELIEVER !!

"I Love it !!. . .It's really a incredible O/S. It' just great! Now I see why people who have seen it say they are now believers. I know I am."
LANCE MICKLUS

1) Large (8") drive support.
2) Double Sided drive support.
3) Double Density drive support.
4) 80 Track drive support.
*NOTE all above drives may be mixed on any one system and can be configured at Sysgen time or during any Backup!
5) Winchester technology fixed drive support.
6) Supports any combination of the above drives up to a max. of 8 drives.
7) Supports double-speed processor clock modifications. (Archbold for example)
8) FASTER! ——— Improved overlay structure using ISAM accessing techniques improves loading times by up to 1400%.
9) General purpose output spoolers of a true, symbiont design provide simultaneous output and program execution without any user intervention.
10) Keyboard Type-Ahead feature permits you to enter keystrokes before your programs need them.
11) User definable keys, all 26 letters.
12) Built in Graphic string packer lets you enter graphic symbols into a BASIC program from the keyboard through the use of the (Clear) key. The (Clear) key is simply held down (just like the (Shift) keys) during other keystrokes and viola...graphics!
13) Dated files. —— All files are accompanied by the date of their last modification (creation or write).
14) Marked files. —— All files are accompanied by a 'mark' if they have been modified since they were last backed up. This permits the BACKUP utility to copy only those files which have actually been updated since a previous backup.
15) File transfer by class. Allows transferring of all files of a similar directory classification such as /CMD, /BAS, /PCL, etc.

VTOS 4.Ø

VTOS 4.Ø
Operating System Diskette with Operator's Guide
$99.95

VTOS 4.Ø
Master Reference Manual
$35.00

VTOS 4.Ø
Combination – 4.0 disk, Operator's Guide, and Master Reference Manual
$125.00

16) Built-in SYSTEM command contains lower case display driver, screen print, break key disable, blinking cursor, disk drive stepping rate and motor-on delay modifications, and more.
17) User may SYSGEN a custom VTOS system configuration containing special I/O drivers, device LINKing and ROUTEing, SPOOLing and DEBUG tasks, etc. which will be automatically loaded during the BOOT process without requiring a more lengthy AUTO and CHAIN procedure.
18) Non-BREAKable AUTO and CHAIN commands.
19) Wild-card DIRectory. Permits you to locate all files of a certain classification such as '/BAS'. Uniformly indicates file size in K (1024 bytes) regardless of drive type. "DIR D" would give you all your files that start with "D".
20) Dynamic file name defaults in APPEND, COPY, and RENAME commands allow you to specify only minimal information about file names.
21) COPY and APPEND commands execute up to 300% faster.
22) ALLOCate command for pre-allocation and non-releasibility of file space. File space will never shrink if this option used.
23) MEMORY command for directly setting upper memory limit.
24) Variable Length file support is incorporated which automatically blocks short user data records both within a sector and across sector boundaries thereby taking maximum advantage of disk file space.
25) No security disk needed to make backups or to run the system!
26) Though many O/S bear his design and code VTOS 4.0 is the only Fully Aproved Operating System by Randy Cook! And it is FANTASTIC!
27) Endorsed by Scott Adams and Lance Micklus!

AUSTRALIAN PRODUCTS

green screen

- Give your monitor the luminous green character found on the very expensive computer.
- Add professional look to your system & programmes.
- The screen may be easily removed; no modification to monitor.
- This product is not an inferior thin product, but 1/8" thick quality material. For business applications, reduced eye strain which improves operator efficiency. For the hobbyist, brighter and bolder graphics.

Screen for Model 1* $19.95
Model 2 $24.95

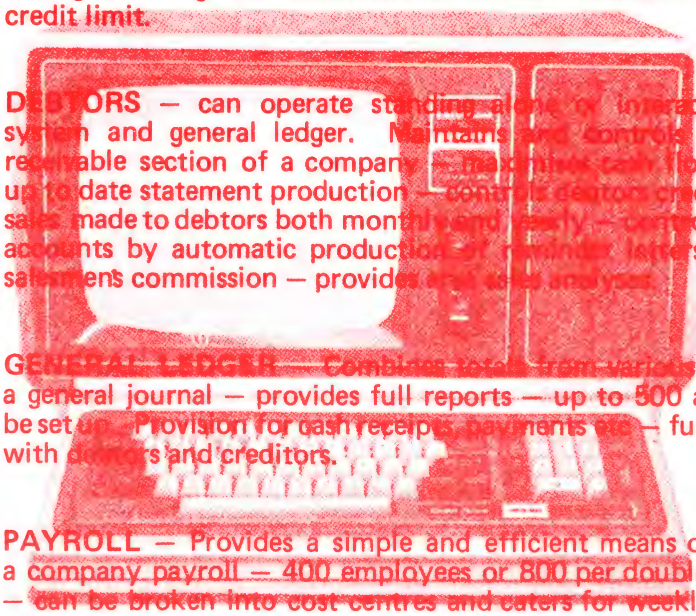
**Specify if current model VDU.*

DISKMASTER organises your disks. This program automatically reads disk directory and even renames disk if you wish and gives either a screen or printer output if your software. Even shows free granules and what systems on each disk. Only required one drive $24.95

AZIMUTH FINDER — Amateur radio gives beam heading, call signs and distance for 100's of countries screen or printer. $14.00

MODEL 2

Deforest Software proudly announce a fine range of fully interactive software written in Australia for Australian conditions.



INVOICING WITH STOCK CONTROL — Will maintain and control stock levels, produce a detailed invoice or credit note on line — will analyse sales and profitibility of all stock lines — analyses sales and profitibility of stock groups. Fully interactive with debtors system — during invoicing shows current stock levels and details and customer credit limit.

$700

DEBTORS — can operate standing alone or interact with stock system and general ledger. Maintains and controls the accounts receivable section of a company — maximises cash flow by prompt up to date statement production — controls debtors credit — analyses sales made to debtors both monthly and yearly — controls delinquent accounts by automatic production of reminder letters — calculates salesmens commission — provides area sales analyses.

$500

GENERAL LEDGER — Combines totals from various journals into a general journal — provides full reports — up to 500 accounts may be set up. Provision for cash receipts, payments etc — fully interactive with debtors and creditors.

$500

PAYROLL — Provides a simple and efficient means of automating a company payroll — 400 employees or 800 per double density disk — can be broken into cost centres and caters for weekly, fortnightly and monthly pay periods — provision for cash includes coinage, cheque and bank transfer — a total of 20 allowances or deductions are provided for either before or after tax — 10 rates of pay.

$700

These programs can be sold separately or as part of a complete computer system, from as little as **$8200** (depending on options).
The lease amount on this figure is only **$45.40** (PER WEEK)

Fully Interactive Business Programmes

Number I in software for the Model II

MODEL II EXTENDED BASIC

A family of enhancements to the Model II BASIC interpreter. Part of the package originated with the best of APPARAT, INC's thoughts in implementing NEWDOS BASIC. The system is written entirely in machine language for SUPER FAST execution. The extensions are fully integrated into Model II BASIC AND REQUIRE NO user memory, and NO user disk space. The package is made up of the following five modules, each of which may be purchased separately.

XBASIC — Six single keystroke commands to list the first, last, previous, next or current program line, or to edit the current line. Ten single character abbreviations for frequently used commands: AUTO, CLS, DELETE, EDIT, KILL, LIST, MERGE, NEW, LLIST and SYSTEM. $25

XREF — A powerful cross reference facility with output to display and/or printer. Trace a variable through the code. Determine easily if a variable is in use. $40

XDUMP — Permits the programmer to display and/or print the value of any or all program variables. Each element of any array is listed separately. $40

XRENUM — An enhanced program line numbering facility which allows specification of an upper limit of the block of lines to be renumbered, supports relocation of renumbered blocks of code, and supports duplication of blocks of code. $40

XFIND — Permits quick and easy location of specified strings or keywords within the program text. $30

SAVE - on the purchase of the entire package
$200

SKRUNCH

A SUPER FAST TRSDOS UTILITY. Compresses your BASIC programs to an absolute minimum. Typically saves 30-40% space, even for programs without REM statements! Also results in 7-10% improvement in execution speed. $35

FRIEND
FOUR NEW TRSDOS COMMANDS!

SHOW — A much better multi-disk directory display. Let's you see only those files you want, and includes date of last update.
MOVE — A much better file copying command. Copy/Move whole groups of files, renaming them at the same time if desired, with just one command!
ERASE — Better than KILL. Better than PURGE.
PRINT — Print BASIC programs from disk, whether saved in ASCII or compressed.
All 4 DOS commands allow fast processing of one, or complete groups of files, based on generic naming and wild card specifications. Enhanced functions too numerous to fully describe here.

EXAMPLES:
SHOW PAY*/BAS:*
Directory display of all'/BAS' files on all diskettes which begin with 'PAY'.
MOVE PAY*/BAS 1 TO /OLD:3
Save current versions of payroll programs to drive 3, changing extension to '/OLD'.
MOVE OLD*/* TO NEW /:1
Copy all files on drive 0 which begins with 'OLD', regardless to extension to drive 1 changing the first three letters of the file name to 'NEW', but retaining the same file extension. Save time!
Reduce frustration!
Eliminate ERROR 33! $75

SPOOLER - Model I and Model II

Our workhorse! This package, available for Model I, in the TRSDOS/NEWDOS or NEWDOS 80 versions, or for the Model II, greatly enhances system performance when running typical business applications. Many applications have been benchmarked to run nearly TWICE AS FAST with the SPOOLER installed. Installs in minutes, and no changes are required to your programs. Preferred Model II versions require NO user memory. Optional features for the Model II version only. Serial printer support, and DISK SPOOLING support is particularly recommended for word processing applications. $100

SERIAL PRINTER OPTION $50
DISK SPOOLING OPTION $50

ITOII

A helping hand when converting BASIC programs from the Model I to the Model II Automatically adjust PRINT @, and PRINT USING to compensate for differences in the language. Advises you where adjustments are necessary for PEEK, POKE etc. $25

HOSTII/TERMII

Allows 'remote control' of a Model II from another Model II, or any ASCII terminal. If terminal is a Model II, accurate screen positioning (PRINT @) is fully supported. Requires NO user memory! This system is designed to provide software support to our customer locations without ever leaving the office. $50

SAVE

Retrieve the resident BASIC program following an accidental re-boot, an accidental SYSTEM or a system crash. DON'T BE WITHOUT THIS ONE. YOU NEVER KNOW WHEN YOU WILL NEED IT! $35

DOSFIX

A collection of patches to TRSDOS and BASIC to enhance their usability and function. Includes our well known BREAK7E patches to keep the break key from being used accidentally. FREE WITH ANY MODEL IF SOFTWARE PURCHASE.

THE FINE PRINT

TRS-80 is a trademark of the Radio Shack division of Tandy Corporation.
NEWDOS and NEWDOS/80 are trademarks of Apparat, Inc.

T DAM
Automatically read and write your random access files. Never worry about fielding again $35

DIVERGE
Compare 2 BASIC program files showing differences. $35

REBUILD
Reorganise programs for adding code and rearranging $35

SIFTER
12 high speed sorts inc. demo programs $35

SHRINK
Make every byte count!
Reduce by 25-40% $35

SUPERSEDE
Write programs in shorthand, change variable names $35

MINGLE 2
Merge up to 14 files. $35

AIDS 3
Marvellous data base c/w machine code assisted sorts and print.
sub system $125

CALCULATION SUB SYSTEM 3
Fields in any order.
Columnar totals and subtotals, computations using field values and constants $46

TRS-80* (Model II)

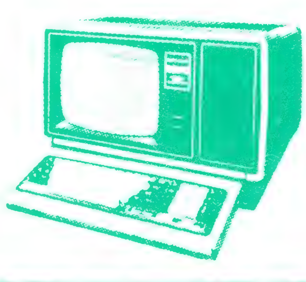
CPM version 2.1. . . . .$210
MAGIC WAND
"The Ultimate Word Processor"
(requires CPM) $450
C Basic 2 . . . .$120

The following business programs are fully interactive and written for Australian conditions.

Creditors $400
Debtors and Invoicing $500
Order Entry/Invoicing/Stock
Control. $700
General Ledger $500
Payroll $700
Direct Mail $700

DSM
Disk Sort Merge for random files — superfast sort $160
Basic Cross Reference Utility — Seek and find instructions for variables, line numbers, strings, keywords $55

GSF
The standard against which other sorts are compared; machine language, multi-key, USR peeks and pokes. $160



Adapter for TRS-80* computer eliminates disk read errors

De Forest Software is marketing a simple plug-in adaptor for TRS-80* computers that corrects a design defficiency in the disk controller circuit.

The problem, which causes disk read errors, has been traced to Tandy's reliance on a circuit internal to the FD1771 controller IC to perform the function of separating clock and data pulses.

As explained in the *Backgrounder*, use of the internal chip circuit for reliable data-clock separation is a design shortcut which the manufacturer of the controller IC warns against.

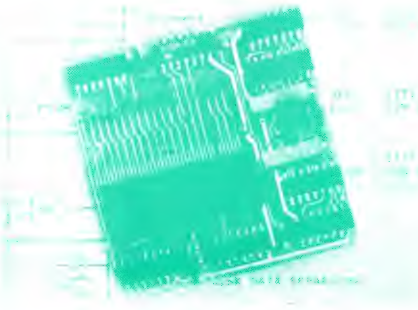
The De Forest solution, a PC card adaptor called the DATA SEPARATOR, eliminates the problem by substituting an explicit data separator circuit — one which has been used reliably in disk controllers since 1977 — for the internal IC separator circuit.

The DATA SEPARATOR is installed without modifying the host system. The user merely removes the FD1771 IC from the host controller, installs the IC in the DIP socket on the SEPARATOR card, and plugs the adaptor into the vacated socket of the host controller and then joins two wires.

We caution that opening the Expansion Interface of the TRS-80* computer which is required to install the SEPARATOR, may void the computer's limited 90 day warranty.

The SEPARATOR which sells for $29.95, may be purchased from De Forest dealers or ordered direct from De Forest Software.

Payment for mail orders may be made by cheque or money order or charegd to a BANKCARD account.



Adaptor fixes TRS-80 computer disk controller.*

CRC ERROR! TRACK LOCKED OUT!

Technical Staff
Percom Data Company

This problem started while we were studying an annoying problem with the TRS-80* computer. Disk drives sold by Percom are realigned and tested before shipment. We noticed, however, that some disk drives would pass the Percom inspection but just would not work reliably on the inner tracks with a TRS-80* computer. These drives were within the manufacturer's specifications, and would function perfectly on other disk systems Percom manufactures — "perfectly" here meaning more than 50 million bytes read without error!

The disk read data separation arrangement in the TRS-80* computer Expansion Interface uses an internal data separator of the FD1771 disk formatter/ controller IC. Use of the FD1771 internal data separator is not recommended by Western Digital, the IC manufacturer. The following note appears on page 17 of the FD1771 data sheet:

Internal data separation may work for some applications. However for applications requiring high data recovery reliability, WDC recommends external data separation be used.

We suspected the data separator because the problem was most severe on disk inner tracks where storage density is highest and data separation is most critical.

To prove our point, a technician breadboarded a standard Percom data separator circuit, and configured it to plug directly into the FD1771 IC socket of the TRS-80* computer controller.

When connected to the TRS-80* computer, a trouble-some drive functioned perfectly! We ran a BACKUP utility many times and never got a track lockout. Before we added the external data separator circuit to the computer, this same drive would always lock out tracks, and would have difficulty reading from the inner (higher number) tracks.

The data separator circuit fixes the mini-disk controller of the TRS-80* computer. The type of drives being used is irrelevant; the circuit eliminates disk read errors resulting from the inability of the Tandy controller design to reliably separate clock and data signals when reading high density inner tracks.

This Australian Unit has been designed and produced for deForest Software **ONLY $29.95**

HIGH RESOLUTION FOR THE TRS-80*

NOT A KIT!

INSTALLS IN SECONDS!

DOES NOT AFFECT WARRANTY!

384 x 194 RESOLUTION

UPPER AND LOWER CASE WITH DESCENDERS

SUPER 80 is a unique Australian invented and manufactured hardware/software package which will integrate high speed, high resolution graphics into any level II TRS-80 system. The installation of **SUPER 80** will not affect normal operation of the TRS-80. A fully buffered expansion and edge connector is provided so that other peripherals may be used. Fully protected against over voltage.

*Comes complete with SOFTWARE
POWER SUPPLY
MANUAL
CABLE

ONLY $439.00

Also available at City Personal Computers, 75 Castlereigh St., Sydney

10 PART PASCAL SERIES

THE COMPLETE PASCAL

BY SUE EISENBACH AND CHRIS SADLER

CHAPTER 6 DATA STRUCTURES 2~RECORDS AND FILES

Computer programmers, the languages they program in and sometimes even the computers on which these programs run tend to be biased either towards number-crunching (immense calculations) or data-processing (huge quantities of information). This chapter is intended to provide an introduction to PASCAL's approach to the second of these.

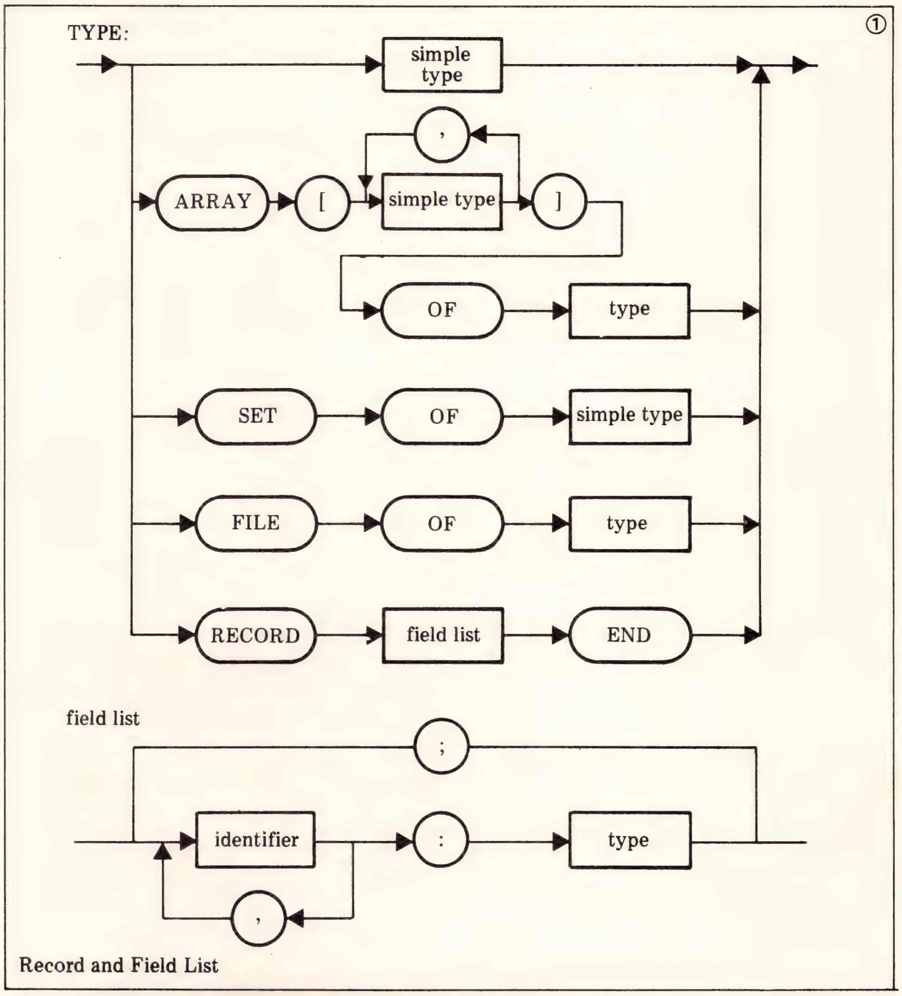
Computers have traditionally been employed in the fields of scientific research and business data-processing. The different requirements of these two types of user have produced opposing specialisms amongst computer professionals — conflicting designs and configurations of both hardware and software; and most importantly from our point of view, programming languages with differing facilities and capabilities. Scientific languages tend to standardize on specialized and sophisticated mathematical functions and to leave non-standard and bulk-data handling features which are consequently provided (with greater or lesser degrees of effectiveness) by the individual implementors of the language. This reflects perfectly reasonably the general format of a mathematical problem where complex operations need to be performed on a relatively restricted amount of data.

Commercial languages, however, often don't provide sophisticated or even convenient mathematical functions since their processing tends to consist of more routine operations but with much larger quantities of data. This is not to suggest that a good sorting algorithm is not every bit as complex as, say, a Fourier transform module, but while the latter operates *on* the supplied data to produce completely different data, the former works *with* data, re-ordering it but not actually changing any values. In any case, in a typical data-processing problem, the quantity of supplied data is generally so large that no more than a small fraction can fit into the machine at one time — the organizational problems associated with containing this data in machine-readable form and of making it available to the program in a controlled and ordered manner dominate these commercial languages.

While the data is being manipulated within the machine it is grouped together in structures called *records*. Loosely, a record is a number of data items, usually of different types, which need to be associated in some way, probably because they all pertain to a single entity. A second record would contain the corresponding information, in the same format, pertaining to another entity, and so on. An entry in a telephone directory, i.e. Name, Address . . . Telephone No. is a simple example of a record.

A *file* is a data structure external to the program and consists of a collection of records. The characteristics of any particular file will depend not only on the size and number of the records it is to contain, but also on the *medium* on which the file is being stored. Magnetic tape files are called *sequential* files because records are stored in sequence and can only be accessed as such — i.e. start at the beginning and deal with each record in turn. Clearly, quite a bit of complicated programming has to be done at system level to control the tape drive and the motion of data through the read/write tape heads. This software can usually be initiated by fairly simple calls embedded in the programming language. Wirth's standard PASCAL provides a set of these sequential file-handling facilities.

PASCAL, however, was designed when discs were considered as a sort of extension of the memory in large computer systems and were too expensive and bulky to be a suitable medium of data file storage. The advent of small hard disc packs and reliable diskettes has put this medium within reach of smaller system users making it reasona-



Record and Field List

②

```
1  PROGRAM FIRSTILL ;
2  TYPE STOCK=RECORD
3               NUMBER:INTEGER ;
4               NAME:PACKED ARRAY [1..24] OF CHAR ;
5               PRICE:REAL ;
6               QUANTITY:INTEGER ;
7               VAT:0..100
8            END ; (*RECORD*)
9  VAR ANSWER:CHAR ;
10      ITEM:ARRAY[0..4]OF STOCK ;
11      ACCEPTABLE:SET OF 'A'..'Z' ;
12      I:-1..4 ;
13 PROCEDURE SETUP ;
14 BEGIN
15     ITEM[0].NAME := 'DAISY BELL PRINTER      ' ;
16     ITEM[1].NAME := 'MICRO DOT MATRIX PRINTER' ;
17     ITEM[2].NAME := 'NCR PAPER               ' ;
18     ITEM[3].NAME := 'CONSTAT PAPER           ' ;
19     ITEM[4].NAME := 'CARBON RIBBON           ' ;
20     FOR I := 0 TO 4 DO
21     BEGIN
22        ITEM[I].NUMBER := I ;
23        ITEM[I].QUANTITY := 0 ;
24        PAGE (OUTPUT) ;
25        WRITELN ('PLEASE TYPE IN TODAY''S PRICE FOR ', ITEM[I].NAME) ;
26        WRITE('FOLLOWED BY THE VAT RATE AS A % -->') ;
27        READ (ITEM[I].PRICE) ; READLN (ITEM[I].VAT)
28     END
29 END ; (*SETUP*)
30 PROCEDURE HELP ;
31 BEGIN
32     PAGE (OUTPUT) ;
33     WRITELN('TYPE H TO SEE THIS DISPLAY.') ;
34     WRITELN('     T TO PRODUCE A TILL SLIP.') ;
35     WRITELN('     S TO PRODUCE A SUMMARY OF THE DAY''S TRANSACTIONS.') ;
35     WRITELN('     E TO EXIT FROM THIS PROGRAM.') ;
36     WRITELN ; WRITELN ;
37     WRITE ('WHEN PRODUCING A TILL SLIP TYPE EACH ITEM NUMBER FINISHING ') ;
38     WRITELN('WITH A -1.') ;
39     WRITE ('HIT THE RETURN KEY TO CONTINUE.') ;
40     READLN
41 END ;(*HELP*)
42 PROCEDURE TILLSLIP ;
43 VAR TOTAL, TAX:REAL ;
44     NUM:INTEGER ;
45 BEGIN
46     TOTAL := 0 ;
47     TAX := 0 ;
48     READLN (NUM) ;
49     WHILE (NUM > -1) AND (NUM < 5) DO
50     BEGIN
51         WRITELN (ITEM[NUM].NAME, '    ', ITEM[NUM].PRICE) ;
52         ITEM[NUM].QUANTITY:=ITEM[NUM].QUANTITY+1 ;
53         TOTAL:=TOTAL+ITEM[NUM].PRICE ;
54         TAX:=TAX+0.01*ITEM[NUM].VAT ;
55         READLN (NUM)
56     END ;
57     WRITELN ;
58     WRITELN('VAT            ',TAX) ;
59     WRITELN('TOTAL          ',TOTAL+TAX) ;
60     READLN
61 END ;(*TILLSLIP*)
62
63 PROCEDURE SUMMARY;
64 CONST TAB = '         ' ;
65 VAR TOTAL,TAX:REAL ;
66 BEGIN
67    TOTAL:=0 ;
68    TAX :=  0 ;
69    PAGE (OUTPUT) ;
70    WRITELN('NAME                              QTY SOLD            AMOUNT') ;
71    FOR I:=0 TO 4 DO
72    BEGIN
73       WRITELN(ITEM[I].NAME, TAB, ITEM[I].QUANTITY, TAB,
74               ITEM[I].QUANTITY*ITEM[I].PRICE) ;
75       TAX := TAX + 0.01*ITEM[I].VAT*ITEM[I].QUANTITY ;
76       TOTAL := TOTAL + ITEM[I].PRICE * ITEM[I].QUANTITY ;
77    END ;
78    WRITELN ; WRITELN ;
79    WRITELN ('SUBTOTAL = ', TOTAL) ;
80    WRITELN ('VAT = ', TAX) ;
81    WRITELN('TOTAL = ', TOTAL + TAX) ;
82    READLN
83 END ; (*SUMMARY*)
84 BEGIN (*MAIN PROGRAM*)
85     SETUP ;
86     ACCEPTABLE := ['E', 'H', 'S', 'T'] ;
87     WRITELN ('TYPE H FOR HELP.') ;
88     REPEAT
89         READLN (ANSWER) ;
90         IF NOT (ANSWER IN ACCEPTABLE) THEN ANSWER := 'H' ;
91         CASE ANSWER OF
92              'E' : WRITELN ('GOOD BYE') ;
93              'H' : HELP ;
94              'S' : SUMMARY ;
95              'T' : TILLSLIP
96         END (*CASE*)
97     UNTIL ANSWER = 'E'
98 END .
```

ble to discuss direct-access files. As with the mag. tape drive, special system software is required to direct the read/write heads to the correct track and sector on the disc and to control the flow of data to and from this location. However, all the data is spread over the *surface* of the disc and is consequently all equally accessible *directly* — hence the name.

Although this software is utilized at operating system level (in the form of file-handling and/or editing utilities), high-level language calls are seldom available to the programmer so that most disc data-files tend to be sequential. UCSD PASCAL is an exception to this general rule and we feel that *direct-access* facilities are sufficiently important to be incorporated in *any* future standard PASCAL. It is with a small degree of reluctance therefore that we abandon Wirth PASCAL in Section 4 to describe the UCSD file-handling facilities.

Records

The record was defined in the previous section as a grouping of associated data items. These data items are known as the *fields* of the record. There is no restriction on the type which each field may be so that the structure is distinct from the array where all elements must be of the same type. In addition, fields are not directly accessible via computable indices like array elements, but must be referenced by a fixed *field identifier.*

The record is declared in a TYPE statement in which is stipulated both the field identifiers and their corresponding types. The syntax diagram in Box 1 shows the reserved words required for this declaration, together with the format for the *field list.* Note that a field within a record could be another record, or even an array.

As an example of the uses of records in a program look at program FIRSTILL in Box 2. The program represents a cash register for a small shop which sells printers and stationery for microcomputers. A tally is kept of every sale so that, in addition to producing a slip for the customer, a daily summary can be output at closing time. The record type STOCK is declared in lines 2 to 8 with the field list laid out in lines 3 to 7. The field NAME is declared as a PACKED ARRAY. *Packing* is a device whereby elements of a particular data type, are packed into the smallest amount of memory needed — e.g. a bit for a BOOLEAN, a byte for a CHAR etc. Numerical array elements frequently incur too large an overhead to make packing worthwhile but BOOLEANs and CHARs usually repay packing with substantial space saving. The PACKED ARRAY OF CHAR is formally defined as a *string* which we shall be dealing with at some length in the next chapter.

In line 10, array ITEM is declared as of type STOCK which implies that 5 records will be set aside in memory for this data structure. Each record can be referenced by a different value of the array index. Line 15 and the rest of procedure SET UP provide illustrations of the method by which individual fields within a record are referenced. The record name and the field name, separated by a . , must both be supplied, and lines 15 to 19 refer to the same field in different records. Lines 22 and 23 on the other hand refer to different fields in the same record (selected by I). The instruction in line 24 clears the screen (in UCSD PASCAL).

Procedure HELP reveals the menu-driven nature of the program, since each of the different functions may be selected by inputting a single character at the keyboard. The most important key to remember, especially for an inexperienced teller, is 'H' which executes HELP itself. The two procedures TILLSLIP and SUMMARY show how record fields can be manipulated like ordinary variables although the referencing scheme makes them appear a bit long-winded. This can be avoided by means of the WITH statement whose syntax diagram is given in Box 3. When the record identifier is given in the "variable" box, all identifiers appearing in the "statement" are checked by the compiler against the field names pertaining to that record as well as the normal declared identifiers appropriate to that procedure. The record name is thus taken as a *default* for the duration of the statement. This is illustrated in the new version of SUMMARY appearing in Box 4, lines 10 to 15.

Exercise: Re-write FIRSTILL using WITH statements where appropriate.

NOT DESIGNED FOR MUSIC

THE TROUBLE WITH TRS-80's

We like Tandy's home computer. At 16K Level II stage it is a reliable and reasonably priced entry to the magic of computers. But that cassette!

Tandy's optional "glitch kit" makes it fairly dependable, but it's still slow, requires knob twiddling for volume level and button pushing for operation.

Enough to put you off computers!

TANDY's ANSWER

Of course Tandy has an answer, add an expansion interface and a small disk drive.

Ask the price!

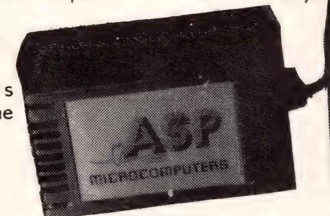
ASP's ANSWER

Now we have an alternative, and its not just a matter of offering a non-standard small disk drive a little cheaper than Tandy with the attendant risk if it doesn't work.

STRINGY FLOPPY

The solution is a compact high-speed tape transport system entirely under the control of the computer - designed for data - not music. Its called STRINGY FLOPPY

There are no controls to fiddle with, just one light indicating the drive is in operation and another indicating that data is being written onto wafer. It simply plugs into the back of the Tandy keyboard unit. Or with an optional adaptor into the System 80 Computer sold by Dick Smith.



WAFERS - NOT CASSETTES

Unlike the cassette recorder that comes with the Tandy, STRINGY FLOPPY uses specially designed "wafers" about the size of a credit card and 5 mm. thick. These contain an endless loop of special chromium dioxide tape ranging in length from 1.6 to 23 metres. Their low mass means they operate reliably at high speed.

Removal of a reflective label protects them against accidental overwriting of data.

Their special tape and the digital recording method used means you don't lose data.

RELIABLE RECORDING

STRINGY FLOPPY uses a true digital recording technique like floppy disks, not fluctuating audio tone. The result - RELIABILITY.

The system was designed for computers - not designed for music.

NO MAINTENANCE DRIVE UNIT

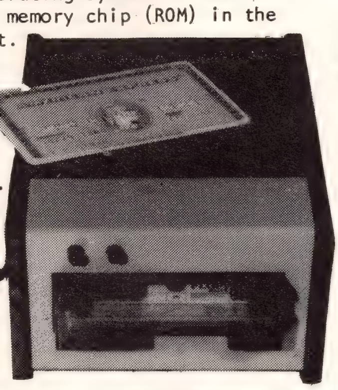
The STRINGY FLOPPY wafer has the pressure roller for the capstan built in. So the capstan and the record head can be fixed in the die-cast aluminium drive. No adjustments required. The wafer just slides into the drive and "clicks" home. No other mechanical motion is required. And because wafers contain endless loops, the tape always travels in the same direction.

There is no need for a rewind capability.

Simplicity means reliability.

INTEGRATION TO THE TANDY

All this would be useless if STRINGY FLOPPY wasn't easy to use. So its operating system is in a permanent memory chip (ROM) in the drive unit. Using the SYSTEM command it becomes part of the Tandy.



14K of ROM instead of just 12K. Commands allow you to initialize wafers (prove their integrity), load or save up to 99 files on a wafer on up to 8 STRINGY FLOPPY drives connected to the Tandy.

Programs can be in BASIC or machine language. Loading the 1K DATA I/O program from wafer adds additional commands to store and retrieve data rather than programs.

SPECIAL BONUS

Those who know their Tandy's well will know all about the dreaded keybounce! Well, STRINGY FLOPPY's ROM contains a keybounce fix routine. Nice?

RECORDING SPEED DENSITY

STRINGY FLOPPY records at 7200 baud (700 characters per second), 14 times faster than Tandy's cassette on a Level II machine. Tape speed is 25 cms. per second, so you can calculate access and loading times. If you "hot-up" your Tandy by increasing the clock speed you will get a proportionate increase in data density and therefore wafer capacity.

Even on a standard Tandy you can fit up to 64K on a 23 metre wafer.

PROGRAMMING

With STRINGY FLOPPY you can create programs which chain under the control of your computer. The pressure on memory diminishes. ASP has programs such as its word processor and a data base manager which can demonstrate the possibilities for clever creative programming.

THE CATCH

OK, the bottom line. STRINGY FLOPPY complete with operating system, plug pack power supply, DATA I/O program is $350.00 including Sales Tax. Add $5.00 for freight within Australia.

And we'll include a box of 10 mixed length wafers (normally $35.00) and a 2 for 1 bus extender (normally $20.00) in the price.

And a six month limited warranty.

And a year's subscription to our STRINGY FLOPPY Newsletter.

AND if you're not satisfied with STRINGY FLOPPY, return it within two weeks for a courteous refund (this applies only to non-Bankcard purchases).

So maybe there wasn't a catch.

BANKCARD orders must quote full Bankcard number, Expiry Date and be signed.

Versions for other computers are available.

ASP
MICROCOMPUTERS

797 DANDENONG ROAD,
EAST MALVERN 3145.
VICTORIA AUSTRALIA.
Telephone: 211 8855 211 8344

Files

One of the essential characteristics of a file is that it is external to the program as a whole. Only a small portion of the data is accessible to the program at any one time and although it is possible to have a file of arrays, say, we will assume that a typical file contains records. In this section we are discussing the sequential files of Wirth PASCAL as defined in the introduction so that the file will consist of a *sequence* of records in strict order. When a file is accessed therefore, the "unit" in which the program must deal with the data is one record.

A file is declared by means of a type statement as shown in the syntax diagram of Box 1. In our case, the "type" referred to in the declaration will be a record which will have been declared earlier on in the declaration part. When the compiler encounters the file declaration, apart from noting the *file identifier* and establishing the correct I/O channel (and peripheral) on which the file is to be found, it creates a structure in memory of exactly the type (i.e. record) previously defined. This structure is known as the *file window* or *buffer variable* and is referenced as follows;

file identifier ^
or file identifier ↑

depending on the character set supported by your terminal.

During execution of the ensuing program, any reference to "file identifier ↑" will involve those memory locations set aside for that structure. It is the job of the programmer, however, to ensure that the *contents* of these locations are in fact the fields of the record under consideration. For this purpose there are a number of *file-handling operators* available. These enable the programmer to manipulate the peripheral on which the file is stored and so access the data needed.

The file-handling operators are

RESET (filename) — starts at the beginning of the file and puts the first record into the buffer variable. This is used when *reading* data out of a file.

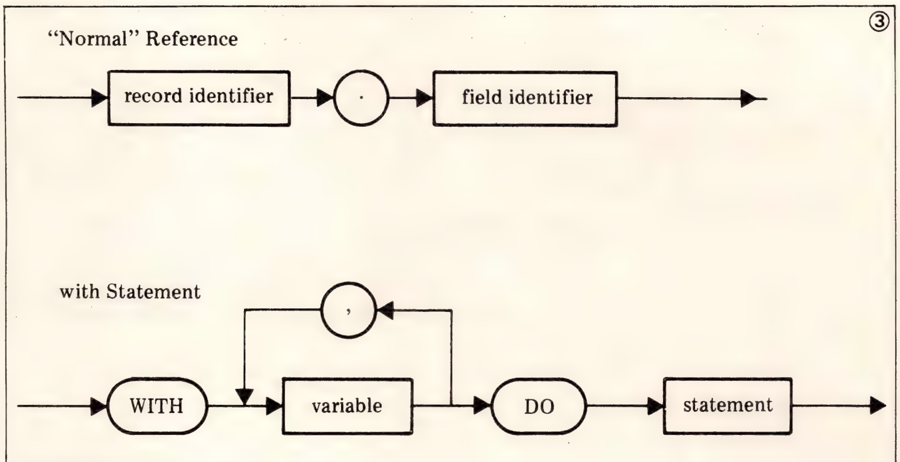
REWRITE (filename) — starts at the beginning of a new file or out-of-date file for the purpose of *writing* to the file. Nothing is actually written on the file at this stage, however.

GET (filename) — advances the file window by one record and assigns the data contained therein to the buffer variable.

PUT (filename) — writes contents of buffer variable out to file — i.e. creates a new record at the end of the file.

In addition to the file window, another file control element is maintained in the machine while file operations occur. This is a BOOLEAN variable called EOF (for end-of-file) which is FALSE as long as there are unaccessed records still in the file and becomes TRUE when the last record is reached. When a RESET is executed, EOF is made FALSE unless no file can be found. When a REWRITE is executed EOF is made TRUE. A GET won't work unless EOF is FALSE beforehand and a PUT won't work unless EOF is TRUE beforehand. This makes it impossible to write a record into the middle of a file.

PROGRAM BIGTILL in Box 5 is an



Field Referencing

④

```
PROCEDURE SUMMARY ;
CONST TAB = '          ' ;
VAR TOTAL, TAX: REAL ;
BEGIN
    TOTAL := 0 ;
    TAX := 0 ;
    PAGE (OUTPUT) ;
    WRITELN ('NAME                        QUANTITY SOLD       AMOUNT') ;
    FOR I := 0 TO 4 DO
        WITH ITEM[I] DO
        BEGIN
            WRITELN(NAME, TAB, QUANTITY, TAB, QUANTITY*PRICE) ;
            TAX := TAX + 0.01*VAT*QUANTITY ;
            TOTAL := TOTAL + QUANTITY*PRICE
        END ; (*WITH*)
    WRITELN ; WRITELN ;
    WRITELN ('SUBTOTAL = ', TOTAL) ;
    WRITELN ('VAT = ' , TAX) ;
    WRITELN ('TOTAL = ', TOTAL + TAX) ;
    READLN
END ; (*SUMMARY*)
```

⑤

```
PROGRAM BIGTILL ;
CONST MAX = 100 ;
TYPE STOCK = RECORD
                NUMBER : INTEGER ;
                NAME : STRING[25] ; (* UCSD ONLY *)
                PRICE : REAL ;
                TOTQUANTITY : INTEGER ;
                QUANTITYSOLD : INTEGER ;
                REORDERLEVEL : INTEGER ;
                VAT : 0..100
             END ; (*RECORD*)
VAR ANSWER, OLD : CHAR ;
    ITEM : ARRAY [1..MAX] OF STOCK ;
    STOCKFILE : FILE OF STOCK ;
    ACCEPTABLE : SET OF 'A'..'Z' ;
    DAYTAX, DAYTOTAL : REAL ;
    TOTNUM : INTEGER ;

PROCEDURE SETUP ;
VAR I : INTEGER ;
BEGIN
    I := 1 ;
    RESET (STOCKFILE, 'RECORDS.DATA') ;
    WHILE NOT EOF (STOCKFILE) DO
    BEGIN
        ITEM[I] := STOCKFILE^ ;
        I := I + 1 ;
        GET (STOCKFILE)
    END ;
    CLOSE(STOCKFILE,LOCK) ;
    TOTNUM := I - 1
END ; (*SETUP*)

PROCEDURE INITIALISE ;
VAR I, NUM : INTEGER ;
BEGIN
    WRITE('HOW MANY DIFFERENT ITEMS WILL BE SOLD -->') ;
    READLN (TOTNUM) ;
    REWRITE (STOCKFILE, 'RECORDS.DATA') ;
    FOR I := 1 TO TOTNUM DO
        WITH STOCKFILE^ DO
        BEGIN
            NUMBER := I ;
            WRITE('NAME -->') ;
            READLN (NAME) ;
            WRITE ('PRICE -->')  ;
            READLN (PRICE) ;
            WRITE ('STOCKLEVEL -->') ;
            READLN (TOTQUANTITY) ;
            QUANTITYSOLD := 0 ;
            WRITE ('REORDER LEVEL -->' ) ;
            READLN (REORDERLEVEL) ;
            WRITE ('VAT AS A % -->') ;
            READLN (VAT) ;
            PUT (STOCKFILE)
      END ; (*WITH*)
      CLOSE(STOCKFILE,LOCK) ;
      SETUP
END ; (*INITIALISE*)

PROCEDURE WRITEFILE ;
VAR I : INTEGER ;
BEGIN
    REWRITE (STOCKFILE, 'RECORDS.DATA') ;
    FOR I := 1 TO TOTNUM DO
    BEGIN
        STOCKFILE^ := ITEM[I] ;
        PUT (STOCKFILE)
    END ;
    CLOSE(STOCKFILE,LOCK)
END; (*WRITEFILE*)
```

5 cont'd

```
73  PROCEDURE TILLSLIP ;
74  VAR TOTAL, TAX : REAL ;
75      NUM : INTEGER ;
76  BEGIN
77      TOTAL := 0 ;
78      TAX := 0 ;
79      READLN (NUM) ;
80      WHILE (NUM > 0) AND (NUM <= TOTNUM) DO
81          WITH ITEM[NUM] DO
82          BEGIN
83             WRITELN(NAME, '  ', PRICE) ;
84             TOTQUANTITY := TOTQUANTITY - 1 ;
85             QUANTITYSOLD := QUANTITYSOLD + 1 ;
86             TOTAL :=TOTAL + PRICE ;
87             TAX := TAX + 0.01*VAT*PRICE ;
88             READLN (NUM)
89          END (*WITH*) ;
90          DAYTAX := DAYTAX + TAX ;
91          DAYTOTAL := DAYTOTAL + TOTAL ;
92          WRITELN ;
93          WRITELN ('VAT      ', TAX) ;
94          WRITELN ('TOTAL    ', TOTAL + TAX) ;
95          READLN
96  END ; (*TILLSLIP*)
97
98  PROCEDURE DAYSTILL ;
99  BEGIN
100     WRITELN ('SUBTOTAL = ' , DAYTOTAL) ;
101     WRITELN ('VAT =       ', DAYTAX) ;
102     WRITELN ('TOTAL =     ', DAYTOTAL + DAYTAX) ;
103     READLN
104 END ; (*DAYSTILL*)
105
106 PROCEDURE WEEK ;
107 VAR I : INTEGER ;
108 BEGIN
109     WRITELN('NUMBER       NAME          PRICE         STOCK   SOLD  REORDER  VAT');
110     FOR I := 1 TO TOTNUM DO
111        WITH ITEM[I] DO
112        BEGIN
113           WRITE(NUMBER,'               ',NAME,'       ',PRICE,'        ') ;
114           WRITE(TOTQUANTITY, '      ',QUANTITYSOLD) ;
115           IF REORDERLEVEL>TOTQUANTITY - QUANTITYSOLD
116           THEN WRITE ('      Y       ')
117           ELSE WRITE ('      N       ') ;
118           WRITELN (VAT) ;
119           QUANTITYSOLD := 0
120        END ; (*WITH*)
121     READLN
122 END ; (*WEEK*)
123
124 PROCEDURE AMENDFILE ;
125 VAR NUM, FIELD : INTEGER ;
126     CONT : CHAR ;
127 PROCEDURE RECMENU ;
128 BEGIN
129     WRITELN('TYPE 0 FOR NO CHANGES.') ;
130     WRITELN('      1 TO ALTER A NAME.') ;
131     WRITELN('      2 TO ALTER A PRICE.') ;
132     WRITELN('      3 TO ALTER A CURRENT STOCK LEVEL.') ;
133     WRITELN('      4 TO ALTER A WEEKLY SALES LEVEL.') ;
134     WRITELN('      5 TO ALTER A REORDERING LEVEL.') ;
135     WRITELN('      6 TO ALTER A VAT RATE.') ;
136     READLN
137 END ; (*RECMENU*)
138
139 PROCEDURE NOCHANGE ;
140 BEGIN
141     WRITELN (' NO CHANGES MADE. ') ; READLN
142 END ; (*NOCHANGE*)
143 PROCEDURE NAMECHANGE ;
144 BEGIN
145     WRITELN ('OLD NAME -->', ITEM[NUM].NAME) ;
146     WRITE ('NEW NAME -->') ;
147     READLN (ITEM[NUM].NAME)
148 END ; (*NAMECHANGE*)
149
150 PROCEDURE PRICECHANGE ;
151 BEGIN
152     WRITELN ('OLD PRICE -->', ITEM[NUM].PRICE) ;
153     WRITE ('NEW PRICE -->') ;
154     READLN (ITEM[NUM].PRICE)
155 END ; (*PRICECHANGE*)
156
157 PROCEDURE TOTCHANGE ;
158 BEGIN
159     WRITELN ('OLD STOCK LEVEL -->', ITEM[NUM].TOTQUANTITY) ;
160     WRITE ('NEW STOCK LEVEL -->') ;
161     READLN (ITEM[NUM].TOTQUANTITY)
162 END ; (*TOTCHANGE*)
163
164 PROCEDURE SOLDCHANGE ;
165 BEGIN
166      WRITELN ('NUMBER SOLD -->', ITEM[NUM].QUANTITYSOLD) ;
167      WRITE ('NEW NUMBER SOLD -->') ;
168      READLN (ITEM[NUM].QUANTITYSOLD)
169 END ; (*SOLDCHANGE*)
170
171 PROCEDURE ORDERCHANGE ;
172 BEGIN
173     WRITELN ('OLD REORDERING LEVEL -->', ITEM[NUM].REORDERLEVEL) ;
174     WRITE ('NEW REORDERING LEVEL -->') ;
175     READLN (ITEM[NUM].REORDERLEVEL)
176 END ; (*ORDERCHANGE*)
177 PROCEDURE VATCHANGE ;
178 BEGIN
179     WRITELN ('OLD VAT RATE -->', ITEM[NUM].VAT) ;
180     WRITE ('NEW VAT RATE -->') ;
181     READLN (ITEM[NUM].VAT)
182 END ; (*VATCHANGE*)
183 BEGIN (*AMENDFILE*)
184     REPEAT
185         REPEAT
186             WRITE ('RECORD NUMBER -->');
187             READLN (NUM)
188         UNTIL (NUM>0) AND (NUM <= TOTNUM) ;
189         WITH ITEM[NUM] DO
190         BEGIN
191             RECMENU ;
192             READLN (FIELD) ;
193             IF (FIELD > 6) OR (FIELD < 0) THEN FIELD := 0 ;
194             CASE FIELD OF
195               0 : NOCHANGE ;
196               1 : NAMECHANGE ;
197               2 : PRICECHANGE ;
198               3 : TOTCHANGE ;
199               4 : SOLDCHANGE ;
```

expanded version of FIRSTILL. In FIRSTILL the data was input at the beginning of each program run. This may be acceptable for a shop that sells five items, but for one that sells fifty it would be a tedious and time consuming process. BIGTILL differs from FIRSTILL in that the records are held on disc in a file (called RECORDS.DATA), loaded into memory at the start of each day's transactions and copied back at the end of each day. Throughout the day the records are held in memory in array ITEM.

In FIRSTILL, PROCEDURE SUMMARY produced the day's results. In BIGTILL results are produced weekly by PROCEDURE WEEK (lines 106 through 122). As it's important to know what should be in the till at the end of each day PROCEDURE DAYSTILL (lines 98 through 104) is provided. DAYTOTAL (line 225) and DAYTAX (line 226) keep tabs of the shop's money and the government's money respectively.

Upon starting up the execution of the program the user is asked if there is an old file (line 227). If the answer is yes, PROCEDURE SETUP (lines 19 through 32) opens the file (line 23) and gets the first record. Note that RESET takes two parameters — the identifier STOCKFILE and the string RECORDS. DATA (which actually appears in the system directory). The second parameter is required by UCSD PASCAL and is not required in standard PASCAL. In lines 24 through 29 each record is read, one at a time, from the STOCKFILE into ITEM. The loop is terminated when the End of File marker is hit (line 14). Line 30 contains another reserved word, CLOSE, that is needed only in UCSD PASCAL. In this version of PASCAL files must be closed before the next RESET or REWRITE can occur. CLOSE(X) deletes X as well as closing it while CLOSE(X, LOCK) retains X in the directory.

If the user does not have a file, then PROCEDURE INITIALISE (lines 34 through 59) is called. In line 39 the STOCKFILE is opened for writing. (Note that RECORDS.DATA is only needed by UCSD PASCAL.) For each record, the FOR DO loop (lines 40-56) reads each field into a record STOCKFILE↑ and then writes this record (line 55) to STOCKFILE. Since this process does not put the information into ITEM it is necessary to call SETUP (line 58) to read the new discfile into memory.

PROCEDURE WRITEFILE (lines 61 through 71) opens the STOCKFILE for writing (line 64) and then in the FOR DO loop (lines 65 through 69) assigns each element of the array ITEM into the file window STOCKFILE↑ so that it can be written to the file (line 68). In fact, line 67 (and line 26) shows one of the major advantages of having a record data structure. Assignment of one record to another of the same type can be done in a single statement. This is true even if the fields of the record contain records, sets and arrays.

PROCEDURE AMENDFILE (lines 124 through 209) allows the user to alter any of the information in array ITEM. This allows for the correction of mistakes made, as well as for changing the stockfile levels when stock comes into the shop or "walks". In line 208

5 cont'd

```
200                 5 : ORDERCHANGE ;
201                 6 : VATCHANGE
202               END (*CASE*)
203
204         END (*WITH*) ;
205         WRITE ('MORE CHANGES, TYPE Y OR N -->') ;
206         READLN (CONT)
207       UNTIL CONT = 'N' ;
208       WRITEFILE (*NOT ESSENTIAL BUT DONE TO MINIMIZE EFFECTS OF A SYSTEM CRASH*)
209   END ; (*AMENDFILE*)
210
211   PROCEDURE HELP ;
212   BEGIN
213       PAGE (OUTPUT) ;
214       WRITELN ('TYPE H TO SEE THIS DISPLAY') ;
215       WRITELN ('     T TO PRODUCE A TILL SLIP') ;
216       WRITELN ('     D TO PRODUCE THE DAY''S TILL TOTALS') ;
217       WRITELN ('     W TO PRODUCE A SUMMARY OF THE WEEK''S SALES') ;
218       WRITELN ('     E TO EXIT FROM THIS PROGRAM') ;
219       WRITELN ('     A TO ALTER THE STOCKFILE') ;
220       READLN
221   END ; (*HELP*)
222
223   BEGIN (*MAIN PROGRAM*)
224       ACCEPTABLE := ['A', 'D', 'E', 'H', 'W', 'T'] ;
225       DAYTOTAL := 0 ;
226       DAYTAX := 0 ;
227       WRITE ('DOES A FILE ALREADY EXIST. TYPE Y OR N  -->') ;
228       READLN (OLD) ;
229       IF OLD = 'Y'
230       THEN SETUP
231       ELSE INITIALISE ;
232       WRITELN ('TYPE H FOR HELP.') ;
233       REPEAT
234           READLN (ANSWER) ;
235           IF NOT (ANSWER IN ACCEPTABLE) THEN ANSWER := 'H' ;
236           CASE ANSWER OF
237               'A' : AMENDFILE ;
238               'D' : DAYSTILL ;
239               'E' : BEGIN WRITEFILE ; WRITELN ('GOOD BYE') END ;
240               'H' : HELP ;
241               'T' : TILLSLIP ;
242               'W' : WEEK
243           END (*CASE*)
244       UNTIL ANSWER = 'E'
245   END .
```

⑥

```
 1  PROCEDURE TILLSLIP ;
 2  VAR TOTAL, TAX : REAL ;
 3      NUM : INTEGER ;
 4  BEGIN
 5      TOTAL := 0 ;
 6      TAX := 0 ;
 7      READLN (NUM) ;
 8      WHILE NUM > -1 DO
 9      BEGIN
10          SEEK (STOCKFILE, NUM) ;
11          GET (STOCKFILE) ;
12          WITH STOCKFILE^ DO
13          BEGIN
14              WRITELN (NAME, '  ', PRICE) ;
15              QUANTITYSOLD := QUANTITYSOLD + 1 ;
16              TOTQUANTITY := TOTQUANTITY - 1 ;
17              TOTAL := TOTAL + PRICE ;
18              TAX := TAX + 0.01*VAT ;
19              SEEK (STOCKFILE, NUM) ;
20              PUT (STOCKFILE) ;
21              READLN (NUM)
22          END (*WITH*)
23      END
24  END ; (*TILLSLIP*)
```

Look up table

Computer jargon
File-handling
Sequential File
Direct Access File
Records
Fields
Master File
File window — Buffer Variable
Update
Packing
Peripheral
Default

PASCAL Reserved Words
RECORD
WITH — DO
FILE — OF
RESET
REWRITE
GET
PUT
EOF

UCSD Excpetions
See Sections 3 & 4
SEEK
CLOSE

Exercises
(i) Rewrite FIRSTILL using WITH
(ii) Rewrite BIGTILL using SEEK

PROCEDURE WRITEFILE is called to make the changes permanent. It isn't essential to do this, since before exiting from the program for the day, the file is written to disc (line 239); it's a precaution to prevent the loss of data if the system crashes.

Compared with handling ordinary variables, the business of file-accessing is clearly rather awkward in programming terms. In particular, where large files of textual materials are concerned, PASCAL supports a number of specialized features. These will be dealt with in our subsequent chapter on word-processing.

Direct Access file handling

Up to this section all the examples have dealt with sets of data that could be completely held in main memory while processing occurred. With memory prices decreasing generally and the new 16-bit micros with their enormous address spaces coming on the market, many applications will actually be able to keep their data in main memory in this way. However, if one isn't planning to purchase a Z8000 with a megabyte of RAM there probably will come a time when the amount of data required is too large for the memory available. In this case files are kept on disc (or tape) and only the record currently being processed will be in memory. As access speeds on disc are very much slower than those of main memory, every effort has to be made to minimize access time.

When data is held in main memory, the data can be updated during each transaction. When the data is held in sequential files, however, such alteration is more complicated. The file must be copied over into a new file, one record at a time. When the record to be altered is reached, it is brought into memory, amended and then written out into the new file. The rest of the file is then transferred as before. Although this technique ensures that the data being accessed is always up-to-date, the delay between transactions would be of the order of minutes for any reasonably sized file. In consequence, sequential files are not usually updated in this way. Instead, a *secondary* file with the update information is established and all alterations over some period (e.g. a day) are collected. At the end of the period the *master* file is updated. Unfortunately, as this period drags on, the master file becomes progressively more inaccurate and in some applications (e.g. airline reservation systems) such out-of-date information is completely unacceptable.

If PASCAL is to become acceptable as a viable language for data processing, it will have to offer the more convenient direct-access facilities associated with disc-based backing store rather than the current standard tape-based sequential access methods. We hope that the standards bodies currently working on PASCAL will take this into account. In the meantime we have taken the liberty of discussing the UCSD implementation of these features which, although non-standard, are widely available on micros.

SEEK is a UCSD reserved word that will search out an individual record from a disc file. SEEK requires two parameters, the first being the file identifier, and the second, an integer representing the record number to which the window must be moved. The first record of a UCSD direct-access file is number Ø.

If STOCKFILE in program BIGTILL became so large that the internal array ITEM could not fit into the available memory, several changes would be necessary in the program. Since only one record would be present in memory, the array ITEM would become superfluous. Procedure TILLSLIP in BOX 6 is a rewrite of the version in Box 6. Line 10 locates the required record while line 11 reads it into the window STOCKFILE^ STOCKFILE in line 12 corresponds to ITEM[NUM] of line 81 Box 5. After the information has been accessed and altered (lines 14-18) the amended record is copied back into STOCKFILE. Line 19 is necessary because a GET moves the window forward one record, so that PUT in line 20 would otherwise overwrite the (NUM + 1)th record rather than the NUMth.

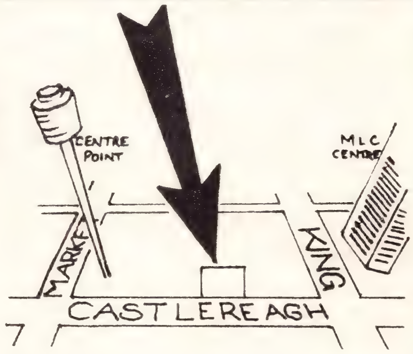
Exercise: Re-write BIGTILL for a direct access master file.

Conclusion

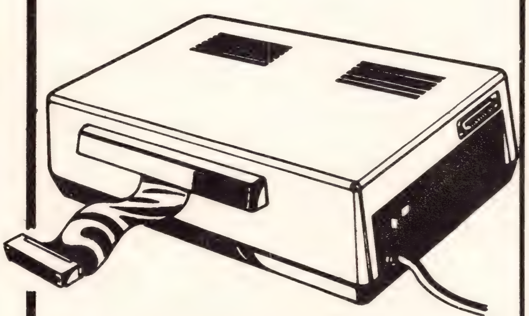
Different methods of file-access and their relation to the different media on which the information is stored have been discussed. It would be misleading to pretend that "normal" data processing programs are as trivial as the examples we have discussed, but we hope that they have been sufficiently realistic at least to illustrate the concepts involved.

City Personal Computers

75 CASTLEREAGH STREET,
SYDNEY
BOX 4715 G.P.O. SYDNEY 2001



SYSTEM 80 S-100 EXPANSION INTERFACE



- Includes controller for up to four single density disk drives.
- Centronics type parallel port.
- RS 232 C Serial communications port.
- Two vacant S-100 card slots.

ALL FOR $499.00

16K EXPANSION CARD for System 80 Expansion Interface.
Just plugs in and gives you extra 16K of RAM with room for another 16K on the same board. . . . **$199.00**

Further 16K RAM . . . **$ 59.95**

DAISY WHEEL PRINTER at only **$1995** allows word processing at super-low cost. Come in and talk to us about it.

City Personal Computers is just a couple of months old and yet already we are becoming established as Sydney's leading computer shop.

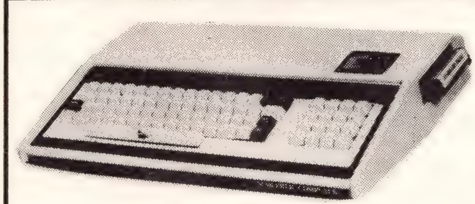
We're right in the heart of the City, so we're easy to get to, and when you reach us you can choose from the widest selection of computer equipment and software to be found anywhere.

Just look at some of our suppliers:
APPLE, COMMODORE, SINCLAIR, DICK SMITH, DEFOREST, EXATRON, MICROPOLIS, PERTEC, PROGRAMMA, SYSTEM SOFTWARE . . .

And the list goes on and on.

By offering the widest range of hardware and software we aim to make computing more enjoyable for you.

So why not call in soon and see what we have to offer.



EXIDY SORCERER
with 8K BASIC ROM Pac.

SORCERER CORNER

We are now stockists of software from such highly respected houses as SYSTEM SOFTWARE of Perth and SOFTWARE SOURCE of Sydney. Some examples of our range:

SYSTEM 2 by Richard Swannell
Features include:
Screen Editor, Function key operators, Renumbering routine, BASIC buffer protector, Printer driver, Revival routine and many others . . . **$ 25.00**

RS 232 PRINTER DRIVER
can be added to SYSTEM 2 . . **$ 10.00**

GALAXIANS the ultimate 'invader' game for the Sorcerer . . . **$ 19.95**

MACHINE CODE TUTORIAL PACKAGE, set of 8 tutorial programs on cassette . . . **$ 24.95**

SOFTWARE SOURCE ROM PAC modifications for Sorcerer . . . p.o.a.

HARDWARE:

Sorcerer II 16K RAM . . . **$1,395.00**
S-100 Expansion Bus . . . **$ 575.00**
Micropolis Disk with Controller . . . **$1,350.00**

MICROLINE 80

IMPACT PRINTER

Few printers are as well known and respected as the **MICROLINE 80** and we are offering them at a very attractive:

$895.00
(+ $10 p+p)

Features include:
- Full ASCII upper and lower case
- Full graphics
- 80 or 132 characters per line
- 80 characters per second

Serial and parallel interfaces, tractor feed and connecting cables are available.

SEND TO **City Personal Computers** Box 4715 G.P.O. Sydney, 2000, N.S.W.

ORDER FORM:

Quantity	Item Description	Item Price	Total
		TOTAL	$

BANKCARD/
AMEX NO:..

NAME:.. CARD EXPIRY DATE: ....

ADDRESS:..

.... Postcode.............. SIGNATURE:.......................................

ZX 80 WINNER

Despite the limited time available for submission of entries to our second Sinclair ZX80 competition, "One-liners" arrived thick and fast at our offices in time for the judging.

After some deliberation between the adjudicators, it was decided an entry which printed alphanumeric characters in an infinite loop was not really what was required (despite a claim by the author that, given enough time, the program would generate all possible "One-liners" — and in most languages!)

Gerard Kohne emerged victorious and his entry will be published next month.

As promised, below are the winning entry from the first competition and five runners-up who will each receive a complimentary subscription to APC.

Microprocessor Simulation

by John Tonkin (still on sedatives a month after receiving his ZX80)

John's simulation is an excellent introduction to machine language programming and, even if the subject is not pursued, it provides some understanding of registers and data movement within microprocessors. Computer Science courses in secondary schools might consider such simulations as a way of introducing students to assembly and machine language programming without involving them in the complexities of actual processors.

This program is designed to simulate a simple base 10 (decimal) microprocessor. The microprocessor contains 16 data registers and one accumulator, each of which can contain a value in the range —999 to +999.

The microprocessor has 12 instructions which enable the user to manipulate numerical data between the registers and input and output data.

Programs may be written by using up to 50 program lines each containing an instruction or data.

INSTRUCTIONS:

Central to the microprocessor is the accumulator. Most of the microprocessor's 12 instructions are involved in transferring values to and from the accumulator, from the data registers, and from the keyboard and monitor. The data registers act as memory for the microprocessor.

The instructions are divided into four main categories:

— INPUT/OUTPUT
— DATA CONTROL
— MATHEMATICAL FUNCTIONS
— COMMANDS RELATING TO FLOW OF CONTROL.

Each instruction has a numerical code which must be used in the program.

INPUT/OUTPUT:

INPUT — is used to transfer a number from the keyboard to the accumulator during execution of the program.
NUMERIC CODE : 9

OUTPUT — is used to transfer a number from the accumulator to the monitor.
NUMERIC CODE : 10

DATA LIST — this instruction performs the same function as a DATA LIST (see monitor). It lists the contents of the data registers.
NUMERIC CODE : 11

DATA CONTROL:

LOAD — loads the accumulator with the contents of the data register specified in the next line of the program.
NUMERIC CODE : 4
e.g. 1:4 2:15 will load the accumulator with the contents of data register 15.

STORE — loads the data register specified in the next line of the program with the contents of the accumulator.
NUMERIC CODE : 5

DATA — loads the accumulator with the contents of the next line of the program.
NUMERIC CODE : 6
e.g. (program line numbers 4 and 5) . . .
4:6 5:574 these lines will load the accumulator with the value 574.

MATHEMATICAL FUNCTIONS:

ADD — adds the value of the data register specified in the next line of the program to the accumulator and stores the result in the accumulator.
NUMERIC CODE : 7

MINUS — subtracts the value of the data register specified in the next line from the value of the accumulator and stores the result in the accumulator.
NUMERIC CODE : 8

FLOW OF CONTROL:

JUMP — causes the program to jump to the line number specified in the next line of the program. i.e. the next line to be executed is the one specified.
NUMERIC CODE : 1
e.g. (program line numbers 5 and 6) . . .
5:1 6:31 these lines will cause the program to jump to line 31.

JUMP IF O — causes the program to jump to the line specified if the value of the accumulator is O.
NUMERIC CODE : 2

JUMP IF >O — causes the program to jump to the line specified if the value of the accumulator is greater than O.
NUMERIC CODE : 3

HALT — ends the program and returns control to the monitor.
NUMERIC CODE : 12

MONITOR:

The monitor enables the user to list, alter, and run programs and consists of five commands. When the monitor is run, it will respond with: P/D/L/A/R? This is a request to the operator to enter the appropriate letter for the required command of the following:

P-PROGRAM — enables the operator to enter a program by asking for the line number and contents of the line to be entered. e.g. P/D/L/A/R? P
LINE;? 1
? 9 will enter code 9 into line 1 of the program.

When finished entering the program, enter the next line and then 9999 as its contents to return to the monitor, e.g. LINE ? 43
? 9999
P/D/L/A/R?

D-DATA — enables entry of data directly into the data registers. When executed the monitor requests the number of the register to be altered and then the new contents of that register.
e.g. P/D/L/A/R? D
REG ? 1
? 36 this loads data register 1 with the value 36. To return to the beginning of the monitor enter O as the register number.
e.g. REG ? 4
? 138
REG ? O
P/D/L/A/R?

L-LIST — lists the program and then returns to the beginning of the monitor.

A-LIST DATA — lists the contents of the 16 data registers and then returns control to the beginning of the monitor.

R-RUN — runs the program and then returns control to the beginning of the monitor when a HALT instruction or error is encountered. When an overflow error occurs the monitor prints E.

SUMMARY:

INSTRUCTION	CODE	NUMBER of LINES
Input	9	1
Output	10	1
Data List	11	1
Load	4	2
Store	5	2
Data	6	2
Add	7	2
Minus	8	2
Jump	1	2
Jump if O	2	2
Jump if > O	3	2
Halt	12	1

SAMPLE PROGRAM:

This program is designed to multiply two positive numbers together.

LINE	INSTRUCTION	CODE
1	Input	9
2, 3	Store 1	5, 1
4	Input	9
5, 6	Store 2	5, 2
7, 8	Data — 1	6, —1
9, 10	Add 2	7, 2
11, 12	Store 2	5, 2
13, 14	Jump if O, 23	2, 23
15, 16	Load 1	4, 1
17, 18	Add 1	7, 1
19, 20	Store 1	5, 1
21, 22	Jump 7	1, 7
23, 24	Load 1	4, 1
25	Output	10
26	Halt	12

Enter each line and its respective code. When run the program will ask for two inputs and then print out the product of the two numbers.

Programs with more lines can be written if the computer has more memory.

```
  1 DIM A(50)
  2 DIM B(16)
  5 M=Ø
 10 PRINT "P/D/L/A/R";
 20 INPUT C$
 30 IF C$= "P" THEN 100
 40 IF C$= "D" THEN 200
 50 IF C$= "L" THEN 300
 60 IF C$= "A" THEN 400
 70 IF C$= "R" THEN 5ØØ
 80 GOTO 10
100 PRINT "LINE;";
110 INPUT I
120 IF I > 50 THEN 100
130 INPUT A(I)
140 IF A(I)=9999 THEN 10
150 IF A(I)<999 AND A(I)>-999
    THEN 100
160 PRINT "E"
170 GOTO 130
200 PRINT "REG;";
210 INPUT J
220 IF J<1 OR J>16 THEN 5
230 INPUT B(J)
240 IF B(J)<999 AND B(J)>-999
    THEN 200
250 PRINT "E"
260 GOTO 230
300 FOR K=1 TO 50
310 PRINT K; " " ; A(K) ;":",
320 IF A(K)<>9999 THEN NEXT K
330 GOTO 10
400 FOR T = 1 TO 16
410 PRINT T;" ";B(T),
420 NEXT T
430 IF M=11 THEN RETURN
440 GOTO 5
500 L=0
510 L=L+1
520 IF X>999 OR X<-999 THEN 710
530 IF L>50 THEN 710
540 LET M=A(L)
550 LET N=A(L+1)
560 IF M=1 THEN LET L=N
570 IF M=2 THEN IF X=0 THEN LET L=N
580 IF M=3 THEN IF X>0 THEN LET L=N
590 IF M=4 THEN LET X=B(N)
600 IF M=5 THEN LET B(N)=X
610 IF M=6 THEN LET X=N
620 IF M=7 THEN LET X=X+B(N)
630 IF M=8 THEN LET X=X-B(N)
640 IF M=9 THEN INPUT X
650 IF M=10 THEN PRINT X
660 IF M=11 THEN GOSUB 400
670 IF M=12 THEN GOTO 5
680 IF M>8 THEN 510
690 IF M>3 THEN L=L+2
700 GOTO 520
710 PRINT"E"
720 GOTO 5
730 END
```

Basic E Calculator

by P. Sutherland

This program uses a method of evaluation first described by A.H.J. Sale in Computer Journal Vol. II pp 229-230. It is a variation on the standard summing of a power series method and therefore depends on the fact that, except for the first two digits, each successive term and the sum of each successive term is less than one. Therefore, if a truncated portion of the series is multiplied by 10, it will then consist of an integer portion and a fractional remainder series. The integer can be removed and printed, and the process is then repeated, giving in sequence the fractional digits of e.

Since only integer arithmetic is required, except for the calculation of the number of iterations required, this procedure is fairly fast. Given a maximum integer size of 32767 (as on the ZX-80), this algorithm allows the series to include all terms up to 1/32761. In other words if you had the time and the memory, you could calculate e on the ZX-80 to about 10000 digits before losing accuracy.

(Although the author did present a compressed version of the program which was easily within the 1k limit, we present below a fully expanded version for the sake of readability–Ed.)

```
010  REM A program to calculate e to
     as many places as required
020  REM program by P. Sutherland
     20-Sep-1980
030  P=3.14159
040  PRINT "Number of places to
     calculate e to";
050  INPUT N
060  REM This is where we calculate
     the number of iterations
070  REM required for the inner loop,
     as well as the storage
080  REM needed for the coefficient
     matrix C(M).
090  REM we must do at least four
     iterations!
100  M=4
110  T=(N+1)*2.302
120  IF M*(LOG(M) —1) +.5*LOG
     (2*P*M) ) T THEN 160
130  M=M+1
135  GOTO 120
140  REM if there is insufficient mem-
     ory for the number of places
150  REM requested, then BASIC will
     give an error at this point.
160  DIM C(M)
170  REM this is where the real crunch-
     ing is done
180  REM firstly we must initialize the
     coefficient matrix C(M)
190  FOR J=2 TO M
200  C(J)=1
210  NEXT J
220  REM the first place is known!
230  PRINT "e=2.";
240  REM let the computation really
     begin
250  FOR I=1 TO N
260  D=0
270  FOR J=M TO 2 STEP -1
280  T=INT (C(J) *10 + D)
290  D=INT (T/J)
300  C(J)=INT (T—D*J)
310  NEXT J
320  REM after each calculation, we
     output one digit of e!
330  PRINT D;
340  REM and back around for another
     one!
350  NEXT I
360  REM we finish it off nicely!
370  PRINT
380  END
```

Learnit

by D. Traverso

Above all else this program is an exercise in artificial intelligence; it demonstrates in very simple terms how a computer can possess that most basic of human capabilities i.e. the ability to learn.

The program is divided into two parts — the first part being the learning phase, the second being the testing phase. It asks the operator how many geometric shapes it is required to learn, and then one by one the operator must enter the various shapes. They are entered in a 5 by 5 matrix, using full stops as markers for lines. The "1" appearing before and after each line of data is required to allow leading spaces to be entered and also to help remember when to correctly press "CR" i.e. only after five characters have been entered.

The computer studies each object in turn and asks what it is called. Then the computer asks the operator to enter one of the objects which it previously studied. The computer then establishes through the use of logic what the object is called.

The program doesn't simply compare the image with that held in memory, but rather compares the characteristics of the image. This distinction is very important because without it the logic of the computer would suggest that a square is not a square at all simply because it differs in size to that which it previously studied.

This type of program would also be ideal for teaching children the distinguishing features of shapes. The child could share the learning process with the computer and use the testing procedures to test itself as well as the computer.

SAMPLE ENTRY

```
ENTER AN OBJECT PLEASE
? 1.    1
? 1..   1
? 1. .  1
? 1.  . 1
? 1.....1
THIS OBJECT HAS 3 UNEQUAL SIDES

WHAT IS THIS OBJECT CALLED? TRIANGLE
```

```
10 PRINT"'LEARNIT'":PRINT:DIMA(6,6),B(10,11)
20 INPUT"HOW MANY OBJECTS TO LEARN";S
30 FORM=1TOS:GOSUB380 :GOSUB240
40 B(M,10)=A:IFB/A=HTHENB(M,11)=1
50 PRINT"THIS OBJECT HAS"A;:
   IFB/A=HTHENPRINT"EQUAL";:GOTO70
60 PRINT"UNEQUAL";
70 PRINT" SIDES"
80 PRINT:INPUT"WHAT IS THIS OBJECT CALLED";B$
90 FORI=1TOLEN(B$):B(M,I)=ASC(MID$(B$,I,1)):NEXTI
100 NEXTM:PRINT"STUDY COMPLETE":PRINT
120 PRINT"I'VE NOW LEARNED THE FOLLOWING:":PRINT
130 FORI=1TOS:GOSUB410 :NEXTI
140 PRINT:PRINT"NOW TEST ME - CR TO END"
150 GOSUB380 :GOSUB240
160 FORI=1TOS
170 IFB(I,10)<>ATHEN200
180 IFB/A=HANDB(I,11)=1THEN220
190 IFB/A<>HANDB(I,11)=0THEN220
200 NEXTI
210 PRINT"ALIEN OBJECT - TRY AGAIN":GOTO150
220 PRINT"THIS IS A ";:GOSUB410
230 GOTO150
240 B=0:A=0:H=0:FORI=1TO5:FORJ=1TO5
250 IFA(I,J)=46THENC=I:D=J:GOTO270
260 NEXTJ:NEXTI
270 RESTORE
280 FORT=1TO8
290 READE,F
300 IFA(I+E,J+F)=46THEN330
310 NEXTT:RETURN
320 DATA0,1,1,1,1,0,1,-1,0,-1,-1,-1,-1,0,-1,1
330 P=1:IFE<>0ANDF<>0THENP=1.5
340 B=B+P:A(I,J)=36:I=I+E:J=J+F:G=G+1
350 IFA(I+E,J+F)=46THEN340
360 IFG>HTHENH=G
370 G=0:A=A+1:A(C,D)=46:GOTO270
380 PRINT:PRINT"ENTER AN OBJECT PLEASE"
390 FORI=1TO5:INPUTA$:FORJ=1TO5
400 A(I,J)=ASC(MID$(A$,J+1,1)):NEXTJ:NEXTI:RETURN
410 FORJ=1TO9:PRINTCHR$(B(I,J));:NEXTJ:PRINT:RETURN
OK
```

Contact

by A. Moore

This program is designed to help people who are severely physically handicapped but have normal intelligence. The condition is not uncommon among victims of cerebral palsy (spastics). They often cannot speak and random movements of most of their muscles make it very difficult for them to communicate or for their intelligence to be recognised. Fortunately, there is usually some place (thumb, toe, eyelid, etc.) over which they have some control and therefore can work some sort of on-off contact. This could be a push button, interruption of a light beam, a strain gauge to detect a wrinkling of the skin, or any of several other devices. CONTACT allows the output from the device to be interpreted as a message on the screen.

The enclosed version of CONTACT will work on a TRS-80 Level 2. An interface box gives the values 14 or 15 to port 63 when a microswitch is on or off. The screen shows the letters of the alphabet in six groups and a cursor moves past the groups. A group is selected by closing the contact and then the characters of the group are displayed across the screen. A single letter is selected when the cursor passes the letter which is then added to the message being accumulated at the top of the screen. Facilities are available for adjusting the speed of the cursor and for correcting mistakes.

The program can be made more complex in several obvious ways, but when handicapped people are familiarizing themselves with the program, it is more suitable in its simple form. The listing below contains a modification so that pushing the space-bar can simulate the push-button. Should you require input from a microswitch, replace the underlined portion of line 90 with

```
C=INP(63):IF D=C OR C=15 THEN D=C:GOTO95
ELSE D=C:IF D=14 THEN
```

```
5 'FROM ALAN MOORE
10 DIM AL$(6,6):DEFINT R:CLS
15 PRINTSTRING$(7,26)STRING$(22," ")"CON
TACT":PRINTSTRING$(7,26)"ENTER CURSOR JU
MP TIME (SEC)--":INPUT TI:RM=INT(TI*75):
CLS:PRINTCHR$(28)CHR$(23):D=15
20 FOR RW=1TO6:FOR CL=1TO6:READ AL$(RW,C
L):NEXT:NEXT:PRINTCHR$(29)STRING$(9,26)"
XTI  EOR  ASU  NDY  LWV  MBJ  ?? HF   CP
   GX   KZ   Q,   .#  ??"CHR$(28);CHR$(2
3);
25 CX=16149:CY=191:CZ=7:GOSUB 85:IF RQ=1
 THEN 25
35 IF RA=7 THEN PRINT CHR$(8);:GOTO30 EL
SE RD=RA:FOR X=1TO6:Y=16258+10*(X-1):POK
E Y,ASC(AL$(RA,X)):NEXT
45 CX=16322:CY=140:CZ=6:GOSUB 85:IF RQ=1
 THEN 45
50 A$=AL$(RD,RA):IF A$="*" THEN PRINT" "
;:GOTO 55 ELSE IF (A$="?") OR (A$="#"):G
OTO 55 ELSE PRINTA$;
55 FOR RH=16258TO16308 STEP10:POKE RH,32
:NEXT:GOTO 25
60 DATA"*","T","I","H","F","?","E","O","
R","C","P","?"
65 DATA"A","S","U","G","X","?","N","D","
Y","K","Z","?"
70 DATA"L","W","V","Q",",","?","M","B","
J",".","#","?"
85 FOR RA=1TOCZ:IF RA=1 THEN RS=25 ELSE
RS=0
90 RS=RS+RM:RB=CX+10*(RA-1):POKE RB,CY:R
B=RB-10:POKE RB,32:FOR RC=1TORS:C$=INKEY
$:IFC$<>" "THEN95 ELSE RB=RB+10:POKE RB,
32:D=14:RQ=0:RETURN
95 NEXT:NEXT:RB=RB+10:POKE RB,32:RQ=1:RE
TURN
105 POKE RB,CY
```

Mastermoo

by P. Moody, R. Gleeson and M. Lee

The object of the game is the same as that of the well-known "Mastermind". The exception being that instead of colours, numbers are used. The four digit number to be guessed is selected at random, is non-repeating and does not begin with a 0. The player has ten guesses in which to correctly deduce the number, the program awarding a "C" for a correctly guessed digit and a "B" for a correctly guessed digit that is also in the right place. (A "C" stands for a Cow and a "B" stands for a Bull, hence the name "MOO".)

```
10 DIM A(7)
210 LET A(0) = RND(9)
220 FOR J=1 TO 3
230 LET A(J) = RND(10) - 1
240 NEXT J
300 GOSUB 3000
310 IF I=0 THEN GOTO 210
350 FOR I=0 TO 3
360 LET A (I+4) = A(I)
370 NEXT I
375 CLS
380 FOR Z=1 TO 10
400 PRINT "MOO:";
410 PRINT "?";
420 INPUT G
425 IF G < 1 THEN GOTO 815
430 IF G <1023 OR G> 9876 THEN GOTO 410
433 LET A=G
435 GOSUB 4000
440 GOSUB 3000
450 IF I = 0 THEN GOTO 410
455 PRINT A;
500 LET C=0
510 LET B=0
520 FOR I=0 TO 3
530 FOR J=4 TO 7
540 IF NOT A(I)=A(J) THEN GOTO 590
550 IF 4 +I=J THEN GOTO 580
560 LET C=C+1
570 GOTO 590
580 LET B=B+1
590 NEXT J
600 NEXT I
700 PRINT
715 IF B=0 THEN GOTO 745
720 FOR I=1 TO B
730 PRINT "B";
740 NEXT I
745 IF C=0 THEN GOTO 780
750 FOR I=1 TO C
760 PRINT "C";
770 NEXT I
780 PRINT
800 IFB=4 THEN GOTO 870
810 NEXT Z
815 PRINT
820 PRINT "GIVE UP? IT WAS";
821 FOR I=4 TO 7
822 PRINT A(I);
823 NEXT I
825 PRINT
830 PRINT "MORE? (Y/N)"
840 INPUT Q$
850 IF Q$ = "Y" THEN GOTO 210
860 GOTO 9999
870 IF Z=1 THEN GOTO 900
880 PRINT "YOU DID IT IN";Z;"GUESSES"
890 GOTO 830
900 PRINT "***FLUKE***"
910 GOTO 830
3000 FOR I = 0 TO 3
3010 FOR J = 0 TO 3
3015 IF I = J THEN GOTO 3050
3020 IF NOT A(I) = A(J) THEN GOTO 3050
3030 LET I = 0
3040 GOTO 3070
3050 NEXT J
3060 NEXT I
3070 RETURN
4000 FOR I = 0 TO 3
4010 LET A(3-I) = G-(G/10)*10
4020 LET G = G/10
4030 NEXT I
4040 RETURN
9999 CLEAR
```

Dental Calculator

by Dr. J. Fisher

This program, written on an Apple II is designed to help dentists provide patients with accurate quotes for proposed services.

The price of each service is initialized as a variable in lines 120 to 230 and the much used strings initialized in lines 240 to 310. The program could be made to loop back to line 330 but as it stands now it has to be rerun for each patient.

```
100  DIM A(3),C(2),L(3),O(2)
110 A(1) = 18
120 A(2) = 21
130 A(3) = 23
140 C(1) = 19
150 C(2) = 21
160 E = 18
170 G = 14
180 P = 1
190 A$ = " AMALGAM "
200 C$ = " COMPOSITE "
210 E$ = " EXTRACTION"
220 F$ = "FILLINGS"
230 H$ = "HOW MANY "
240 S$ = " SURFACE"
250 P$ = "PINS "
260 N = 0
270 R - 0
280  FOR I = 1 TO 3
290  PRINT H$;I;S$;A$;F$
300  INPUT L(I)
310  IF L(I) = 0 THEN 350
320  PRINT H$;P$
330  INPUT M
340 N = N + M
350  NEXT I
360  FOR I = 1 TO 2
370  PRINT H$;I;S$;C$;F$
380  INPUT O(I)
390  IF O(I) = 0 THEN 430
400  PRINT H$;P$
410  INPUT M
420 N = N + M
430  NEXT I
440  PRINT H$;E$
450  INPUT X
460  PRINT : PRINT " TOTAL CHARGES "
470  FOR I = 1 TO 3
480  IF L(I) = 0 THEN 530
490  PRINT L(I);"  ";I;S$;A$;F$
500 Q = L(I) * A(I)
510  GOSUB 800
520 R = R + Q
530  NEXT
540  FOR I = 1 TO 2
550  IF O(I) = 0 THEN 600
560  PRINT O(I);"  ";I;S$;C$;F$
570 Q = O(I) * C(I)
580  GOSUB 800
590 R = R + Q
600  NEXT
610  IF N THEN  PRINT N;"  ";P$
620 Q = N * P
630  GOSUB 800
640 R = R + Q
650  IF X = 0 THEN 720
660  PRINT X;"  ";E$
670 Q = X * E
680  IF Q < 5 THEN 700
690 Q = 4 * E + (X - 4) * G
700  GOSUB 800
710 R = R + Q
720  PRINT : PRINT " TOTAL: $";R
730  END
800 Q =  INT (Q * 100) / 100
810  HTAB (20): PRINT "$";Q
820  RETURN
```

LEISURE LINES

With J.J. Clessa

The Search for the smallest palindrome with an even number of digits attracted over forty replies. As usual the complaint was "Too Easy", although I didn't receive any analytical solutions to support this view. Instead I had the usual collection of program listings for computers, ranging from main frames, through micros down to programmable calculators. I'll not be publishing any of them, but suffice to say the correct answer is 698896 — which is the square of 836.

I made a draw for the winning entry using Sean Howard's biased random number generator — which should by rights have awarded the prize to one of his relatives. However, it seems to have hit a bug this time . . . unless Heather and James Eaton of Tasmania turn out to be very distant cousins!

Congratulations to the Eatons . . . a Paper Mate pen and pencil set will be winging its way to you c/o the PO — let me know if it ever arrives. Meanwhile it's time to think of more diabolical problems — try this one for size:

PRIZE PUZZLE

This one is quite feasibly done by logical deduction — you could, however, computerise your efforts.

A child's cube has coloured faces. Five colours are possible — red, green, blue, yellow and orange. Three views of the same cube are shown here, and in each of the views the colour on the bottom face of the cube is not repeated on any other face. Which colour occurs twice on the cube ?

Answers please on a postcard to Puzzle No. 7, Australian Personal Computer, P.O. Box 115, Carlton, 3053. All solutions must arrive by January 23rd.

PRIZE FOR THIS MONTH

This month I'm giving away a $20 McGills Newsagency book voucher to the sender of the first correct card out of the bag — good luck!

BLUDNERS

Quite a few readers were thrown into confusion by the ZX80 Breakout program. The problem lies in line 130, at the end, in fact; instead of a space between the inverted commas there should be a shaded square. And in lines 140 and 320, there's only a single space between the inverted commas.

OPTIMIZING TRS-80 EXECUTION TIME

One of the main criticisms levelled against Tandy's TRS-80 pocket computer is its very slow execution speed. Denis Andrews offers some ways of overcoming this handicap.

The TRS-80 stands alone in its professional role, having the flexibility of a computer with the size, immediacy and accuracy of a calculator. However, there will be many others like myself who will wish to explore its full power for satisfaction and amusement.

For programs of more than a few lines, one is likely to encounter unexpected delays much longer than the time required for calculation. The handbooks — which are excellent in most respects — give little guidance about this and are even misleading. This article sets out to provide the missing information.

One program is given to illustrate the points. The chosen problem is fun in its own right — to have the machine play noughts and crosses with variety and skill. This task stretches the machine's logical and memory capacity to the full (and stretched the author too). My first solution took around 70 seconds per move: opponents became impatient! The program below responds in five to 19 seconds, similar to a human player. This has the same logical structure as the first idea, but is now geared to the strengths and weaknesses of the TRS-80.

Strenths and weaknesses

These terms are used in respect to timing. The machine is remarkably powerful and easy to program, but execution is slow. The programmer's staple control devices — FOR loop and GOTO — are the slowest to execute. The less familiar logic functions and arithmetic GOSUB provide faster control. The TRS-80 allows the construction of logic functions using the logic operators =>, <, >=, <= and <>. Such functions take the value 1 when true and 0 when false. The AND and OR operators don't exist in its Basic but their function is performed by * and +. So (B>2)*(B<5) takes the value 1 if B lies between 2 and 5, otherwise it's 0. The familiar devices INT, ABS and SGN are extremely fast.

Table 1 shows the execution time for most logical and arithmetic operations. These were calculated using a FOR loop and stopwatch, adding instructions in various combinations. The values are approximate and disguise marginal interdependencies. (Some are given below.) Times are given both in msec and 25 msec units (bracketed). The latter unit aids rapid calculation of execution times, as the most frequently used words or word-pairs then have unit time.

To calculate actual execution times, you need two further facts:

1. Addresses are accessed by upward search through memory, starting either at the current line or at the first line in memory; the address defines which of these searches is made. GOTO is not costly if it names a current or following line, or line 1. The cost accrues if you have to jump back and thereby search through most of the program. Address searches run at about 450 words per second. FOR and GOSUB work differently: they leave an explicit address in an internal stack and no search is required for the return jump.
2. Variables are also accessed serially but in a complex way. On the right of an assignment (fetch) A is fast and Z slow; on the left (store) the reverse is true. Trading A for Z makes a 16 msec difference. For subscripted variables on the right, there are two fetches (symbol and its reference); the time differences add. On the left, there are two fetches and one store; two cancel, leaving just the symbol, so (A) is fast and (Z) is slow. For practical calculation, reckon each symbol reference one unit, and brackets one or ½ each (left and right of '='). The choice of variable name only matters if frequencies of fetch and store are markedly different. The 'flexible' variables take no longer than the fixed ones: A(26) and A(27) take about the same time in any role.

The time required to jump over most code is around 1/8 to 1/4 of the time needed to execute it! This makes the layout of long programs and their subroutines absolutely crucial. All jumps should be forwards or to early addresses. If you need to jump backwards, create a logically unnecessary GOSUB — for the sake of its stacked address (eg L47 in the program).

Stacked functions seem to occupy a fixed time; the fourth level GOSUB takes no longer than the first, and does not slow the earlier orders either. Likewise for FOR loops. Unexpectedly, GOTO D, GOSUB D, A(D) are not observably slower than GOTO 7, GOSUB 7, A(7), etc., when letter and digit correspond. One letter can be faster than two digits. There is no measurable time added in respect of LET, STEP N or addresses, though presumably they must use some.

The programs provided in the applications manual have not been optimised in either space or time. Most of them have long jumps and make no use of the economical logic functions, the equivalence C=A(3), or the implied '*' (which often saves a pair of brackets as well).

In particular, subroutines are commonly placed at the end, so that the whole program must be jumped to access them. The only references to layout (note 7 on the introductory page) gives the wrong reason for putting subroutines at the start! It would of course be easy to rewrite the programs optimally if one used them often.

Useful devices

1. If a result is needed in two forms, store it twice. Transformations of arrays are slow. (In the program A—I, J—R and T (by itself) represent the entire game. Two of these are redundant images, but transformation would be expensive in time and space.)
2. Data can be packed (L.7, L.70) or unpacked (L.3) in one variable. Decoding is slow, but the device allows much data to be preset. (L.71 is equivalent to 19 assignments +five IF statements.)
3. Multiply rather than divide (L.3, L.59), saving 50 msec.
4. Use IF X. . rather than IF X>5. . , saving 30 msec or more.
5. Make use of the logic functions rather than IF when you can. This generally saves one address and one assignment. (L70 and 71 would be twice the size and run at half speed using IF state-

6 (1/4)	12 (1/2)	25 (1)	36 (1.5)	60 (2.5)	75 (3)	250 (10)
SGN	+!	*	IF	GOSUB N	/	FOR—NEXT
ABS	—	implied*	GOTO N	GOSUB 1		(each
INT	E	(	GOTO 1	(all 4		cycle)
$	"	=	RETURN	levels)		
string	1st digit	)	THEN		FUNCTIONS:	
char.	on LEFT of ASSIGNMENT			√ (11)	DMS (4)	SIN (24)
	Z	N	A	∧ (25)	DEG (5)	ASN (20)
		)		LOG (20)		COS (24)
	(	(		EXP (11)		ACS (23)
	)			LN (25)		TAN(10)
	A	N	Z			ATN(10)
	on RIGHT of ASSIGNMENT					

Table 1 *Approximate execution times, msec (bracketed, 25 msec units).*

ments.)

6. Space saving also saves time; there being less code to jump over.
7. Place subroutines so that they are passed over as seldom as possible.
8. Loop structure is slow however accomplished. Write the inner loop(s) in full if you have space (ends of L.9—17).
9. A fixed string of GOSUBs is appreciably faster than a FOR. . . NEXT loop (and may use less space, too, up to three cycles).
10. Use flexible variables for the ones that must be subscripted anyway. Avoid them if you can.
11. A computation conceived as several successive loops can sometimes be restructured as a single loop with branches (L.9—17 with L.1).
12. Save the early addresses for the most-used subroutines and those which are called from diverse places.

Noughts and crosses

Space is too precious for REM statements in the TRS-80. The following remarks stand in lieu and indicate very briefly how the program works.

L.34 — 47 Set arrays to +1 (no move), mirrored "." for display. Decide who starts. Register first move.

L.4 — 7 Make random choice in range 1—Y. Select 1 of Y digits A(27)+. Record move: machine coded —4, "0"; player —1, "X". S = move no.

L.50 — 63 Display position and machine's response. Read player's move. Transpose first move? Traps are coded only for one of eight symmetries.

L.64—71,L.81,L.84—88 Test for pre-coded traps at machine's second response, (too far from victory or defeat for serial calculation). All wins, losses, alternatives are coded, equivalents played equally often.

L.78 Test for immediate win; enter if any.

L.79 Test for direct threat by player; counter if any.

L.82 Test for empty squares, threats, double threats; select randomly within the highest level registered.

L.9 — 18 Test each remaining move in the rows which contain it; X is criterion: —8(two "0")for win, —2(two "X") to counter, —3("0"+".")for threat.

L.1 The inner routine — only succesful tests enter. Hits listed at A(27)+. If higher-level hit found, restart list and reset level.

This program is capable of further speed optimising, eg by swapping variables. A quite different approach might be better of course. The hybrid of preset and search seems necessary: a fully intelligent program could easily be written but would run extremely slowly. It also seems necessary to avoid duplicating symmetries. One refinement perhaps worth pursuing is to transpose the display instead of the move. My attempts to do this ran out of space, but only just. The listing here includes some 50 redundant words, expended to buy a little speed when the more elegant solution seemed out of reach. A further 70 words would secure the improvement. Any ideas?

```
 1:Z=Z+1:A(Z)=Y:IF U>WLET Z=26:W=U:GOTO 1
 2:RETURN
 4:V=V+9π-50*(V>200):Z=V-INT (V/Y)*Y+1:RETURN
 5:Y=1+INT LOG W:GOSUB 4:W=W*10∧-INT Z:Y=INT ((W-INT W)*10)
 6:A$(Y)="O":X=-4
 7:T=10T+Y:A(Y+9)=X:S=S+1:RETURN
 9:Z=26:IF JLET U=(X=K+L)+(X=M+P)+(X=N+R):IF U>=WLET Y=1:GOSUB 1
10:IF LLET U=(X=J+K)+(X=O+R)+(X=N+P):IF U>=WLET Y=3:GOSUB 1
11:IF NLET U=(X=J+R)+(X=K+Q)+(X=L+P)+(X=M+O):IF U>=WLET Y=5:GOSUB 1
12:IF PLET U=(X=J+M)+(X=N+L)+(X=Q+R):IF U>=WLET Y=7:GOSUB 1
13:IF RLET U=(X=P+Q)+(X=L+O)+(X=N+J):IF U>=WLET Y=9:GOSUB 1
14:IF KLET U=(X=J+L)+(X=N+Q):IF U>=WLET Y=2:GOSUB 1
15:IF MLET U=(X=J+P)+(X=N+O):IF U>=WLET Y=4:GOSUB 1
16:IF OLET U=(X=M+N)+(X=L+R):IF U>=WLET Y=6:GOSUB 1
17:IF QLET U=(X=K+N)+(X=P+R):IF U>=WLET Y=8:GOSUB 1
18:RETURN
34:"G":S=1:T=0:A$(31)="  I GO:"
35:FOR Z=1 TO9:A(Z+9)=1:A$(Z)=".":NEXT Z:INPUT "TOSS (0=YOU START):"
   ;X:IF XLET W=125:GOSUB 5:GOTO 47
44:GOSUB 56:A$(Y)="X":X=-1:GOSUB 7:W=5*(1=T)+158*(2=T)+1379*(5=T)
   :GOSUB 5
47:GOSUB 50:GOTO 47
50:PRINT A$;B$;C$;"/";D$;E$;F$;"/";G$;H$;I$;A$(31);Y:IF -SEND
52:IF S>8 BEEP 3:PRINT "DRAW!":END
54:GOSUB 56:A$(Y)="X":X=-1:GOSUB 7:GOTO 64
56:INPUT "YOUR X:";Y:IF -A(Y+9)PAUSE "FULL":GOTO 56
57:IF T>9RETURN
58:IF (Y=5)+(Y<3)RETURN
59:IF T=5LET Y=Y-2*INT (.5Y-.5):GOTO 63
60:Z=Y/(5-T):IF Z=INT ZLET Y=Y-2:GOTO 63
61:IF T+Y<>8RETURN
62:Y=3
63:BEEP 1:PAUSE "TRANSPOSED..";Y:RETURN
64:IF S=3GOSUB 73-.1T:IF WGOSUB 5:RETURN
65:GOTO 78
67:W=9745-7Y:RETURN
70:W=15*(4=Y)+(7=Y)+4679*(1=Y)+79*(8=Y)+367*(5=Y):RETURN
71:W=37*(9=Y)+457*(2=Y)+479*(3=Y)+357*(6=Y)+349*(5=Y):RETURN
78:IF S>4LET X=-8:W=1:GOSUB 9:IF Z>26GOSUB 6:S=0:A$(31)=" I WIN:"
   :RETURN
79:X=-2:W=1:GOSUB 9:IF Z>26GOSUB 6:RETURN
81:IF S=4GOSUB 90-.01T:IF WGOSUB 5:RETURN
82:X=-3:W=-1:GOSUB 9:Y=Z-26:GOSUB 4:Y=A(Z+26):GOSUB 6:RETURN
84:W=37*(J+R=-5)+19*(L+P=-5):RETURN
87:W=13*((289=T)+(287=T))+1346*((256<T)*(260>T))+47*(213=T):RETURN
88:W=476*(158=T)+238*(156=T)+2468*(159=T):RETURN
90:INPUT V:END:REM -- DELETE LINE AFTER SETTING RANDOM SEED --
```

FACE TO FACE

FREE-FORMAT DIALOGUES

David Hebditch continues his series on the man/machine interface.

Dialogue styles fall into two very broad categories, free-format and formatted (or structured).

We will consider the latter next month and in this article I shall concentrate on free-format interactions between user and system. Most commercial dialogues tend to be highly structured, using formatted screens for data-entry, multi-choice menus, query-by-example and so on. The general view seems to be that the structured approach increases the ease with which the dialogue can be learned and used in practice.

There are no clear definitions which could help us to differentiate between the two categories; the boundary is very fuzzy. But essentially 'free-format' dialogues are those relatively free of syntax and formatting rules. The use of the word 'relatively' is important — nothing can be entirely free of restrictions. Perhaps the key concept is concerned with providing the maximum amount of flexibility in the way in which data can be presented by the user to the system (and vice versa). This contrasts with the structured approach which aims at restricting possibilities.

So different dialogue styles offer users varying *degrees of freedom.* To appreciate this is important; a dialogue has to be designed to offer its most likely users the level of flexibility with which they feel most comfortable. To provide a user with too many possibilities can be as disconcerting for him as offering too few. And, of course, users do vary in their needs; I am sure that after 12 years of working with interactive systems of great variety, I am happier with very open and flexible dialogues than the average first-time user.

If 'free-format' dialogues are not being used too much in regular commercial applications, where *are* they being used? Well, when you type a Basic program into your TRS-80 (or whatever) you are making use of a special-purpose free-format dialogue. The CP/M operating system also employs a relatively 'free' approach; obviously, syntax rules do exist but the program does not control your input on an individual item-by-item (or even character-by-character) basis.

What are the various elements we need to consider in designing a free-format dialogue? These are shown schematically in Figure 2. Each is discussed individually below.

Delimiters

Because free-format input messages are entered as a character string, some convention needs to be employed in order to mark the beginning of one item (or field) and the end of the previous one. In many dialogues, a space may be employed for this purpose but this will obviously be ambiguous when text is being entered. Popular characters for use as delimiters are:

' — ' hyphen
' : ' colon
' / ' oblique stroke
' ( ) ' left and right brackets
' = ' equals
' . ' point
' , ' comma

and so on.

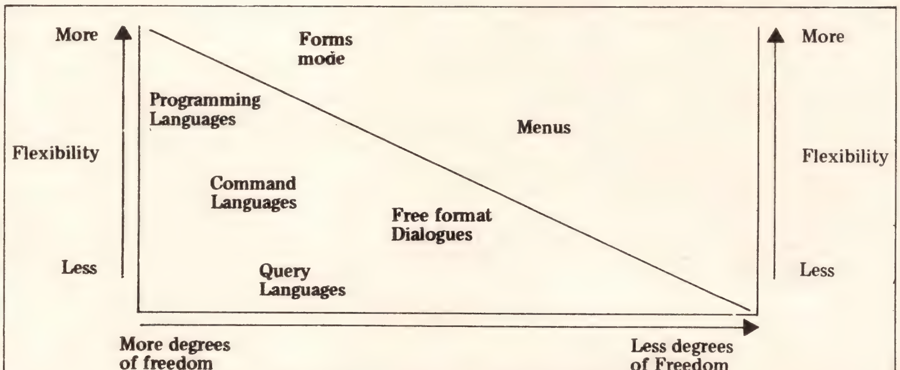


Figure 1 Interactive Dialogues and Degrees of Freedom

A number of factors condition the choice of delimiters:

— avoidance of ambiguity
— ease of location on the keyboard and
— special usage (e.g. the comma used to separate items entered to a Basic INPUT statement).

In many cases, a delimiter can be *implied.* For example, the 15th August 1980 may be entered as:

15.8.80

using the point as a delimiter or as

15AUG80

using the changes from numeric to alphabetic and back again to differentiate the various sub-items. In some cases, a choice delimiter (e.g. ' — ', ' / ' or ' . ' in dates) might serve to make the dialogue more user-friendly

Items

The items which the delimiters frame may be employed for a number of purposes according to the function of the dialogue. Items usually contain:

— commands
— command parameters or, more usually,
— data.

Some examples will serve to illustrate this:

PIP B:PROG1.BAS=A:PROG2.BAS

This is a CP/M PIP utility input line which means "Copy the Basic program file called PROG2 on diskette drive A to diskette drive B and rename it as PROG1." A space delimits the PIP activation command from the rest of the text. Colon (:) prevents the drive designators from being confused with the file names and the point (.) separates the file type subscript from the name. The equals sign (=) is more of a parameter than a delimiter and is best remembered as having the same functional meaning as in a Basic program.

Another interesting feature in this context is that the parameters at each side of the equals sign have positional significance; they will not do the same thing if they were the other way around (quite the opposite!) If it is thought that the positional significance could be a problem, a solution could be the use of 'keywords'.

For example,

PIP FROM=FILE2 TO=FILE1

would mean the same as

MENSA COMPUTERS PTY. LTD.

Suite 3, 454 St. Kilda Road, Melbourne, 3004. Telephone: (03) 26-5683, 26-6150.

FINDEX — The Real Computer

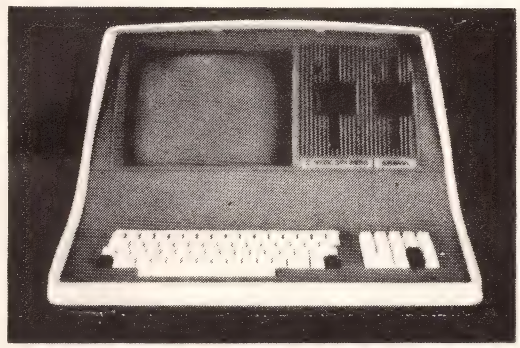
THE WORLD'S FIRST PORTABLE MICROCOMPUTER
Battery or mains operated

RAM 48K to 2 megabytes, bubble memory to 2MB, gas plasma display, optional audio, printer, mass storage mini floppys to 800K bytes, hard disk to 195 megabytes, acoustic coupler, S-100 bus, battery optional, CPU with real time clock. For dynamic businessmen on the move. Ideal for real estate agents, insurance brokers and accountants.

SUPERBRAIN™ SERIES

BUSINESS SYSTEMS

Priced competitively from as low as $45.00 p.w. lease cost including sales tax and software.



Powerful, multi-purpose microcomputer systems.

TYPICAL APPLICATIONS

Debtors ledger and statements, creditors ledger and remittances, general ledger and trial balance, order entry/invoicing, sales analysis, payroll/wages, enquiry, word processing, mailing, record keeping, ledger card, doctors office, real estate, agency accounting, hotel/motel accounting, branch office accounting.
Facilities management consulting.
Software and computer sales.
Computer data preparation.

MENSA G.F. 1000

The Mensa uses a CPU 8080 and Z80 operating at 4MHZ. 64K bytes Dynamic RAM expandable to 2 MB with Bank select allowing multi-user. Texas Instruments 810 Printer. Over 2 MB total 4 drives on-line storage bytes of unformatted data on two double density drives. Optional external hard disk storage can be connected using the optional S-100 Bus. Floppy Disks.

WINCHESTER DISK

26 MB of Winchester Disk complete with controller and easy backup. Disk has special capacity to only back up files accessed during the last period.

Disk Operating System CP/M

OPTIONAL SOFTWARE :

– FORTRAN, COBOL, BASIC

Application Packages: Extensive software development tools are available from leading software vendors, include software for the following applications Payroll, Accounts Receivable, Accounts Payable, Inventory Control, General Ledger and Wordprocessing. All modules mounted to base. CRT in a rigid aluminium frame. Disk Drive assemblies are mounted into special bracket for ease of servicing.

Mensa Computers provide a service network throughout Australia at major service centre locations to minimise response time to service calls. To ensure that equipment will operate at peak performance, engineers and technicians are trained to ensure the highest possible standard of service.

FACE TO FACE

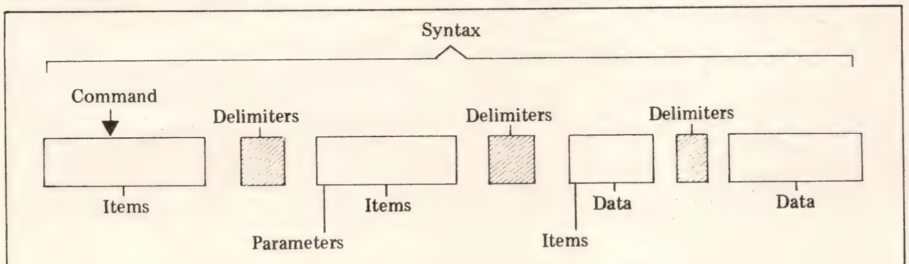


Figure 2: The Elements of a Free-format Message

PIP TO=FILE1 FROM=FILE2

The use of keywords obviously requires more key depressions (*and* you have to remember the words) but where a lot of items could be entered in each, and only a small number are in practice, then the technique might prove to be more economic and reliable in practice. It is also a relatively simple matter to write programs for such formats (especially if the Basic interpreter has an INSTR function).

Free format dialogues can be very economical for simple data entry. For example, to add a new book to a library catalogue file one might type:

0-87626-345-7/GILB & WEINBERG/ HUMANIZED INPUT/WINTHROP/ 1978

In this case, the only 'overhead' characters are the ' / ' delimiters and no coding is employed at all.

Another possibility is to type:

ADD NEW BOOK TO CATALOGUE. ISBN IS 0-87626-345-7, AUTHORS ARE GILB & WEINBERG, TITLE IS HUMANISED INPUT, PUBLISHER IS WINTHROP, YEAR OF PUBLICATION IS 1978.

This is an example of a *natural language dialogue* all of which are 'free format'. Many system designers set out with the good intention of making their dialogues as much like English (or French, Dutch, etc) as possible. This however is much easier said than done; the programming problems alone are enormous. I shall therefore devote a special article to this subject.

A programming technique called 'Finite State Automata' is very appropriate to handling free-format input data and I will write a special article on that topic. In the meantime, here is a checklist of points to keep in mind when designing free-format dialogues:

1. Choose delimiters which are easy to locate on the keyboard (i.e. not in upper case).
2. Avoid possible conflicts in the choice of delimiters.
3. If there are any length limits on items, use prompts of some kind to guide the user.
4. Place optional and/or least used items at the end of the message (so they can be 'dropped off').
5. If the occurrence of items in an input message is low, use a keyword technique or split into a selection of shortened messages.
6. Avoid formats which require awkward keying sequences for non-typists; try typical messages before finalisation.
7. Keep the item sequence as close as possible to that in any related input documents.
8. Minimize the 'overhead' characters and get the computer to use as many cursor/print head functions as possible.

Next month I shall look at structured dialogues.

TRS-80 Tarot

by Simon Williams

This interesting program runs on a 16k machine.

```
10 RANDOM:CLEAR140:GOSUB530
20 DIML(2,10):FORN=0TO9:READL(0,N),L(1,N):NEXT
30 '***INFORMATION/INSTRUCTIONS
40 CLS:PRINT@465,"COPYRIGHT 1980, SIMON WILLIAMS":FORN=1TO1000:NEXT
50 CLS:PRINT"THE TAROT IS AN ANCIENT METHOD OF FORTUNE TELLING WHICH MAKES   USE
 OF A PACK OF 78 CARDS. THESE ARE DIVIDED INTO TWO SECTIONS, THE MAJOR & MINOR '
ARCANAS'. THE MAJOR ARCANA IS A SET OF 22"
60 PRINT"PICTORIAL CARDS DEPICTING PEOPLE AND OBJECTS. THE MINOR ARCANA  IS SIMI
LAR TO AN ORDINARY PACK OF PLAYING CARDS WITH AN EXTRA   CARD IN EACH SUIT, THE
'PAGE'. THE FOUR SUITS ARE NAMED WANDS,"
70 PRINT"CUPS, SWORDS AND PENTACLES. THE MAJOR AND MINOR ARCANAS ARE     SHUFFLE
D TOGETHER AND DEALT OUT, FACE DOWN, IN ANY OF SEVERAL   ARRANGEMENTS KNOWN AS '
LAYS'. THE CARDS ARE THEN REVEALED, ONE"
80 PRINT"AT A TIME, AND READ ACCORDING TO THEIR INDIVIDUAL MEANINGS AND  THEIR P
OSITIONS IN RELATION TO THE OTHER CARDS. I SHALL READ    FROM THE TEN CARD OR 'C
ELTIC CROSS' LAY.
90 PRINT"BEFORE WE START, I NEED TO KNOW SOME THINGS ABOUT YOU, IN ORDER TO ASSI
GN YOU ONE OF THE COURT CARDS AS YOUR 'SIGNIFICATOR'.    ARE YOU MALE OR FEMALE?
 ANSWER M OR F."
100 S$=INKEY$:IFS$<>"M"ANDS$<>"F"GOTO100
110 PRINT@896,"ARE YOU OVER 40? ANSWER Y OR N - PLEASE BE HONEST."
120 A$=INKEY$:IFA$<>"Y"ANDA$<>"N"GOTO120
130 PRINT@896,"WHAT COLOUR IS YOUR HAIR? A = VERY FAIR, B = GRAY, C = LIGHT    B
ROWN, D = DARK BROWN, E = BLACK";
140 H$=INKEY$:IFH$<>"A"ANDH$<>"B"ANDH$<>"C"ANDH$<>"D"ANDH$<>"E"GOTO140
150 A$=S$+A$:IFA$="MY"THENA$="KING OF ":A=14:GOTO180
160 IFA$="FY"THENA$="QUEEN OF ":A=13:GOTO180
170 IFA$="MN"THENA$="KNIGHT OF ":A=12ELSEA$="PAGE OF ":A=11
180 IFH$="A"ORH$="B"THENH$="WANDS":H=64:GOTO210
190 IFH$="C"THENH$="CUPS":H=50:GOTO210
200 IFH$="D"THENH$="SWORDS":H=36ELSEH$="PENTACLES":H=22
210 CD(0)=A+H:CLS:PRINT@320,"THANK YOU. YOUR SIGNIFICATOR IS THE "A$+H$"."
220 PRINT"THIS CARD IS PLACED FACE UP ON THE TABLE AND THE FIRST TWO CARDSOF THE
 LAY ARE PLACED ON TOP OF IT AS THEY ARE DEALT."
230 PRINT@896,"PLEASE CONCENTRATE ON ANY MATTER OF CONCERN BEFORE PRESSING THE S
PACE BAR TO BEGIN THE READING.";
240 IFINKEY$<>" "THENR=RND(0):GOTO240
250 '***READING
260 CLS:PRINT@320,"THE FIRST CARD IS LAID ON TOP OF THE SIGNIFICATOR AND SHOWS T
HE GENERAL SURROUNDINGS OF YOUR ENQUIRY. THE CARD I TURN UP IS ":GOSUB380
270 CLS:PRINT@320,"THE SECOND CARD IS PLACED ACROSS THE FIRST AND INDICATES ANY
   CURRENT PROBLEMS. FAVOURABLE CARDS MEAN SMALL PROBLEMS. I DRAW ":GOSUB380
280 CLS:PRINT@320,"THE THIRD CARD, WHICH INDICATES YOUR AIMS AND IDEALS, IS PLAC
ED ABOVE THE SIGNIFICATOR. IN THIS CASE THE CARD IS ":GOSUB380
290 CLS:PRINT@320,"THE FOURTH CARD, PLACED BELOW THE SIGNIFICATOR, SHOWS THE BAC
K- GROUND TO THE PRESENT SITUATION - WHAT LED UP TO IT. I TURN UP ":GOSUB380
300 CLS:PRINT@320,"LEFT OF THE SIGNIFICATOR, THE FIFTH CARD SHOWS THE IMMEDIATE
   PAST - WHAT IS 'BEHIND' YOU. THIS CARD IS ":GOSUB380
310 CLS:PRINT@320,"THE SIXTH CARD, LAID TO THE RIGHT OF THE SIGNIFICATOR, SHOWS
THEIMMEDIATE FUTURE - WHAT IS 'BEFORE' YOU. THE CARD DRAWN IS ":GOSUB380
320 CLS:PRINT@320,"THE SEVENTH CARD, PLACED TO THE RIGHT OF THE SIXTH, SHOWS YOU
R  ATTITUDE TO THE MATTER. IF A COURT CARD IS DRAWN, YOU ARE SHOW- ING SOME OF T
HE FEATURES OF THAT CHARACTER. I TURN UP ":GOSUB380
330 CLS:PRINT@320,"THE EIGHTH CARD, WHICH INDICATES THE ENVIRONMENT IN WHICH YOU
'REWORKING, IS LAID ABOVE THE SEVENTH. I DRAW ":GOSUB380
340 CLS:PRINT@320,"THE NINTH CARD SHOWS YOUR HOPES AND FEARS AND IS PLACED ABOVE
   THE EIGHTH. THIS CARD IS ":GOSUB380
350 CLS:PRINT@320,"THE LAST CARD, THE TENTH, INDICATES THE FINAL OUTCOME OF THE
   MATTER - WHAT WILL BE - AND IS LAID ABOVE THE NINTH. IT IS ":GOSUB380
360 CLS:PRINT@454,"THIS COMPLETES THE READING. I WISH YOU GOOD FORTUNE.":FORN=1T
O2000:NEXT
370 RESTORE:GOSUB530:PRINT@832,:END
380 '***CARD/MESSAGE SELECTION & PRINT
390 R=RND(156):IFR>78THENN$="     REVERSED":R1=R-78ELSEN$="":R1=R
400 IFR1>64THENR2=R1-64:N$=" OF WANDS"+N$:GOTO450
410 IFR1>50THENR2=R1-50:N$=" OF CUPS"+N$:GOTO450
420 IFR1>36THENR2=R1-36:N$=" OF SWORDS"+N$:GOTO450
430 IFR1>22THENR2=R1-22:N$=" OF PENTACLES"+N$:GOTO450
440 RESTORE:FORN=1TOR1+194:READNM$:NEXT:N$=NM$+N$:MM$="MAJOR":GOTO460
450 RESTORE:FORN=1TOR2+216:READNC$:NEXT:N$="THE "+NC$+N$:MM$="MINOR"
460 IFN1<11THENIFR=CD(N1)THENN1=0:GOTO380ELSEN1=N1+1:GOTO460ELSEN1=0
470 CD(CN+1)=R:RESTORE:FORN=1TOR+230:READT$:NEXT
480 PRINTN$:PRINT"THIS CARD IS OF THE "MM$" ARCANA & IMPLIES "T$
490 FORN=0TO9:SET(L(0,N),L(1,N)):NEXT:PRINT@960,"PLEASE PRESS SPACE BAR TO CONTI
NUE READING";
500 F=F+1:IFF/15=INT(F/15)GOSUB520
510 IFINKEY$=" "THENCN=CN+1:RETURNELSE500
520 IFPOINT(L(0,CN),L(1,CN))RESET(L(0,CN),L(1,CN)):RETURNELSESET(L(0,CN),L(1,CN)
):RETURN
530 '***TITLE GRAPHICS
540 CLS:FORN=1TO10:READC0,C1,C2,C3,C4,C5
550 FORI=C0TOC1:SET(I,C2):NEXTI:FORI=C3TOC4:SET(I,C5):NEXTI,N
560 FORN=1TO6:READC0,C1,C2,C3:FORJ=C0TOC1:SET(C2,J):SET(C3,J):NEXTJ,N
570 FORN=1TO6:READC0,C1,C2,C3,C4
580 FORJ=C0TOC1:SET(C2,J):SET(C3,J):SET(C4,J):NEXTJ,N
590 FORN=1TO5:READC0,C1,C2,C3,C4,C5,C6,C7,C8,C9,CA,CB
600 SET(C0,C1):SET(C2,C3):SET(C4,C5):SET(C6,C7):SET(C8,C9):SET(CA,CB):NEXT
610 FORI=36TO39:SET(I,30):SET(I,35):NEXT:FORI=82TO89:SET(I,25):SET(I,36):NEXT
620 FORJ=11TO13:SET(4,J):SET(5,J):SET(30,J):SET(31,J):NEXT
630 FORI=71TO75:SET(I,25):NEXT:SET(91,16):FORN=1TO1500:NEXT:RETURN
640 DATA48,59,9,6,27,10,46,49,10,32,47,11,16,17,13,6,15,14,46,52,25,44,47,26,38,
53,36,56,57,36,76,79,34,92,95,34,78,81,35,90,93,35,92,99,15,98,101,16,105,111,17
,105,106,35,113,117,35,103,113,36
650 DATA10,15,28,29,14,30,26,27,17,33,24,25,32,34,22,23,33,35,20,21,35,36,18,19
660 DATA31,34,35,36,37,24,35,53,54,55,24,36,62,63,64,29,33,74,75,76,29,33,95,96,
97,9,35,102,103,104
670 DATA38,29,39,29,40,28,41,28,42,27,43,27,65,28,66,28,67,27,68,27,69,26,70,26,
76,28,77,28,78,27,79,27,80,26,81,26,90,26,91,26,92,27,93,27,94,28,95,28,80,36,81
,36,90,36,91,36,112,16,113,16
680 DATA62,7,62,7,62,5,62,9,58,7,66,7,72,7,72,5,72,3,72,1
690 DATATHE MAGICIAN,THE HIGH PRIESTESS,THE EMPRESS,THE EMPEROR,THE HEIROPHANT,T
HE LOVERS,THE CHARIOT,JUSTICE,THE HERMIT,THE WHEEL OF FORTUNE
700 DATASTRENGTH,THE HANGED MAN,DEATH,TEMPERANCE,THE DEVIL,THE TOWER,THE STAR,TH
E MOON,THE SUN,JUDGEMENT,THE WORLD,THE FOOL
710 DATAACE,2,3,4,5,6,7,8,9,10,PAGE,KNIGHT,QUEEN,KING
720 DATAEITHER THE NEED FOR  OR THE SKILLS OF DIPLOMACY.,MYSTERY & THE FUTURE OR
 A WOMAN WITH PSYCHIC POWERS.,FRUITFULNESS IN ALL  ITS ASSOCIATIONS.INITIATIVE.,
STABILITY IN LIFE.   THE AID OF A POWERFUL PERSON.
```

PROGRAMS

```
730 DATAASSOCIATIONS WITH    OTHERS AND PARTICULARLY MARRIAGE.,ALL FORMS OF LOVE
 ANDATTRACTION TO BEAUTIFUL THINGS,TRIUMPH ONLY THROUGH PERSEVERENCE AND DOGGEDN
ESS,THE JUST OUTCOME OF  PROBLEMS.
740 DATACORRUPTION. OTHERS   MAY TRY TO BETRAY YOU.,THE FORCES OF LUCK   AND THE
REFORE SUCCESS.,A TIME FOR TAKING    COURAGE AND ACTING ENERGETICALLY.,INTUITION
. YOUR OWN  OR ADVICE FROM ONE WHO HAS.
750 DATATHE END OF A CYCLE OFEVENTS. CHANGE AND RENEWAL.,A NEED FOR ECONOMY & GO
OD MANAGEMENT.,THE EFFECT OF EXTRA  EFFORT. POSSIBLY VIOLENCE.,ADVERSITY & DISTR
ESS.MISERY CAUSED BY CONFLICT.
760 DATALOSS BY THEFT OF MAT-ERIAL OR SPIRITUAL THINGS.,A WARNING OF HIDDEN  ENE
MIES WHO WILL DECEIVE.,MATERIAL HAPPINESS.  CONTENTMENT THROUGH GOOD MARRIAGE.
770 DATACHANGE OF POSITION.  RENEWAL AND REBIRTH.,ASSURED SUCCESS VIA AVOYAGE OR
 CHANGE OF PLACE.,A TIME TO CONSIDER   YOUR CAREER. CONSOLIDATION.
780 DATAPERFECT CONTENTMENT. ALSO FINANCIAL REWARD.,A TIME FOR GAIETY ANDRECREAT
ION. PERHAPS GOOD NEWS.,CRAFTSMANSHIP AND THENEED TO USE PRACTICAL GIFTS.,GIFT O
R LEGACY. THE  AQUISITION OF NEW POSSESSIONS.
790 DATAMATERIAL DIFFICULTIESBUT LOVE IS WELL LOOKED ON.,GRATIFICATION THROUGHGI
FTS OR OTHER PLEASURES.,INGENUITY IN BUSINESSAND MONEY MATTERS.,A CREATIVE PERIO
D IN CRAFT OR BUSINESS.
800 DATAPRUDENCE. SAFETY AND ACCOMPLISHMENT FROM THIS.,GAIN IN MONETARY OR  FAMI
LY MATTERS.,A TIME FOR SCHOLAR-  SHIP OR REFLECTION. LOOK TO YOUR AFFAIRS.,THE E
FFECT OF SOMEONEWITH YOUR AFFAIRS AT HEART.
810 DATATHE EFFECTS OF AN    OPULENT THOUGH GENEROUS WOMAN. SECURITY.,INTELLECT
AND PARTIC-ULARLY MATHEMATICS. LOOK TO BUSINESS.
820 DATAGREAT FORCE IN LOVE  OR HATE. TRIUMPH THROUGH THIS.,CONFORMITY AND THE
 STABILITY THIS MAY BRING.,A FEELING OF LOSS OF SOMEONE OR SOMETHING.,THE NEED F
OR SOLITUDE. A TIME FOR CONTEMPLATION.
830 DATALOSS OR DESTRUCTION  OF SOMETHING CLOSE TO YOU.,A JOURNEY WHICH MAY  INC
LUDE TRAVEL BY OR OVER WATER.,ANNOYANCE AT THE FAIL-URE OF A PLAN THROUGH DISAGR
EEMENT., BAD NEWS OF AN ILL-  NESS OR OTHER CRISIS.
840 DATADEEP DISAPPOINTMENT  POSSIBLY THROUGH DEATH.,PAIN AND SADNESS. ALLIS NOT
 WELL.,EXAMINATION. THE ABIL-ITY TO STUDY DETAILS NORMALLY MISSED.,SKILL AND BRA
VERY.   THE TIME TO USE SUCH VIRTUES.
850 DATASADNESS OR EMBARRASS-MENT FOR YOU (IF FEMALE) OR A WOMAN CLOSE TO YOU.,T
HE EFFECTS OF ONE   WITH AUTHORITY TO MAKE DECISIONS.
860 DATACONTENTMENT WITH ALL RELATIONSHIPS.,LOVE AND PASSION. THEINTER-RELATION
OF THE SEXES.,A HAPPY CONCLUSION.  MERRIMENT AND FULFILMENT.,TIREDNESS AND IMAGI
N-ARY WORRIES CAUSED BY A STRICT ROUTINE.
870 DATARECEIPT OF A GIFT.   THIS RESULTS IN ANOTHER LOSS.,REFLECTIONS ON THE
PAST AND CHILDHOOD.,IMAGINATION. A GOOD  TIME FOR CREATIVE WORK THOUGH TRANSITOR
Y.,DEJECTION OVER THINGSAS THEY ARE. THIS WILL PASS.
880 DATAA VICTORY IN SOME-   THING YOU DESIRE.,CONTENTMENT WITH YOURPRESENT SURR
OUNDINGS AND THINGS AS THEY ARE.,THE EFFECTS OF A FAIRAND STUDIOUS YOUNG MAN. HE
 MAY HELP YOU.,A PROPOSITION OR IN- VITATION.
890 DATATHE EFFECTS OF A FAIRWOMAN. SHE IS DREAMY BUT VISIONARY.,"THE EFFECTS OF
 A     CREATIVE MAN IN BUSINESS, LAW OR THE CHURCH."
900 DATATHE BEGINNING OF NEW THINGS - CREATION. POSSIBLY A BIRTH.,PHYSICAL SUFFE
RING ORSADNESS. DISSATISFACTION WITH POSSESSIONS.,STRENGTH IN TRADE ANDDEALINGS
WITH PEOPLE.,A HAPPY LIFE. PEACE  AND PROSPERITY.
910 DATAIMITATION OR SHAM. A SUPERFICIAL VIEW.,SUCCESS IN UNDERTAK- INGS OR GREA
T NEWS.,A NEED TO WORK BY DIS-CUSSION AND NEGOTIATION. FINAL SUCCESS.,HASTE TOWA
RDS A FAV- OURABLE OUTCOME. CHANGE.
920 DATADELAY CAUSED BY      OPPOSITION TO YOUR IDEAS.,FINANCIAL GAIN BUT   OPPO
SITION IN LAW.,THE INFLUENCE OF A   DARK YOUNG MAN. HE MAY BRING NEWS.,DEPARTURE
 OR ABSENCE.A MOVE OR EMIGRATION.
930 DATATHE INFLUENCE OF A   COUNTRY WOMAN WHO IS LOVING AND HONOURABLE.,THE INF
LUENCE OF A   FRIENDLY COUNTRYMAN WHO IS HONEST AND CONSCIENTIOUS.
940 DATADISQUIET & A FEELING THAT ALL IS NOT RIGHT.,THE PHYSICAL SIDE OF LOVE. P
ASSION. A VERY STRONG CARD.,LIGHT AND TRUTH IN   MATTERS.  COMPLEX PROBLEMS REVE
ALED.,BENEVOLENCE OF OTHERSTOWARDS YOU.
950 DATAAN INCREASE IN YOUR  UNDERSTANDING OF SOCIETY.,CERTAIN PROJECTS MAY BE F
OOLISHLY THOUGHT OUT.,A DISPUTE. POSSIBLE  LITIGATION.,A TENDENCY TO BIAS.  WATC
H AGAINST BIGOTRY.
960 DATAAN OVERCAUTIOUS ATT- ITUDE. THINK THINGS OUT - THEN ACT.,ABUNDANCE OF TH
E GOODTHINGS IN LIFE. ENJOY THEM.,AN ABUSE OF POWER    THROUGH WEAKNESS.,OBSTRUC
TION THROUGH  SELFISHNESS. TAKE HEED.
970 DATALETHARGY. INACTION   COULD CAUSE PROBLEMS.,CONNECTIONS WITH THE CHURCH O
R RELIGION LOOK WELL.,BLINDNESS TO WHAT IS RIGHT. PETTINESS.,PROBLEMS CAUSED BY
  ARGUEMENT OR DISAGREEMENT.
980 DATAA TENDENCY TO ARRO-  GANCE WHICH MAY CAUSE ALIENATION.,INSTABILITY AND I
N-  CONSTANCY OF THOSE ABOUT YOU.,CONTENTMENT ON MATER-IAL AND SPIRITUAL PLANES.
990 DATAWEAKNESS THROUGH AN  OVERSIMPLIFICATION OF EVENTS.,INERTIA OR FIXITY MAY
CAUSE STAGNATION.,FOLLY THROUGH EXTRA- VAGENCE OR PLEASURE-SEEKING.
1000 DATABAD INTELLIGENCE. THEEVIL SIDE OF WEALTH.,A FACADE OF ENJOYMENTTO KEEP
OTHERS HAPPY.,MEDIOCRITY IN WORK.  A TENDENCY TO PETTINESS.,DELAY TO SOME PROJEC
TDUE TO OTHERS' OPPOSITION.
1010 DATADISORDER OR CHAOS. ITMAY SPELL RUIN FOR SOME PROJECT.,ENVY AND JEALOUSY
    WHICH LEAD TO ILLUSORY DESIRES.,QUARRELS OVER MONEY  OR POSSESSIONS.,A TENDE
NCY TO VANITY.THIS MAY STAND AGAINST AMBITION.
1020 DATADECEPTION. BEWARE OF OTHERS KEEPING BAD FAITH.,MONETARY LOSS THROUGHGAM
BLING OR ROBBERY. TAKE NO CHANCES.,UNFAVOURABLE NEWS    CAUSING WORRY.,IDLENESS
AND LACK OF ENERGY. STAGNATION.
1030 DATASUSPICION AND MIS-   TRUST MAY EFFECT YOUR REASONING.,THE TEMPTATION OF
    VICE. WATCH FOR WEAKNESS.
1040 DATAGREAT FORCE IN LOVE  OR HATE. NOT TO THE  GOOD.,LIES AND DISLOYALTY. BE
WARE UNTRUTHS.,CONFUSION AND A FEEL-ING OF INTELLECTUAL ISOLATION.,PRECAUTION. B
E CARE- FUL WHAT YOU DO.
1050 DATALOSS OR DESTRUCTION  OF SOMETHING CLOSE TO YOU.,A PROPOSAL OF LOVE OROT
HER DECLARATION.PUBLICITY.,GOOD ADVICE FROM A   WISE PERSON. TAKE HEED.,DIFFICUL
TIES AND OPP-OSITION. TREACHERY.
1060 DATASUSPICION CAST ON YOUOR A FEELING OF SHAME.,TRANSITORY PROFIT OR ADVANT
AGE. MAKE HAY...,THE USE OF OBSERVAT- ION TO BAD ENDS. SPYING.,FOOLISHNESS. THE
IN- ABILITY TO THINK SOUNDLY.
1070 DATAA CLOSED MIND. WATCH AGAINST BIGOTRY OR DECEIPT.,A TENDENCY TO CRUELTYO
R PERVERSITY. WATCH THIS.
1080 DATADISCONTENT. THE DE-  SIRE TO PUT RIGHT YOUR PROBLEMS.,LOVE AND PASSION.
 THEINTER-RELATION OF THE SEXES.,A NEW VENTURE OR HOLI-DAY. ACHIEVEMENT FROM THI
S.,NEW RELATIONSHIPS ANDA DELIGHT IN NOVELTY.
1090 DATATHE CHANCE OF NEW    ALLIANCES. BUSINESS LOOKS WELL.,THE CHANCE OF FRES
H  EXCITEMENTS. RENEWAL.,A DESIRE TO SUCCEED. YOUR FORCE OF WILL.,GREAT JOY AND
HAPPI- NESS. A CELEBRATION.
1100 DATALOYALTY OF OTHERS TO YOU. HONESTY AND GOOD ADVICE.,INDIGNATION AT AN
 AFFRONT IN LOVE. A VIOLENT RESPONSE.,A TASTE FOR RISK     COULD LEAD TO DECEPTI
ON.,A TRICK OR FRAUD. BE ON YOUR GUARD.
1110 DATATHE INFLUENCE OF A   DISTINGUISHED WOMAN. DON'T TRUST HER.,THE EFFECT O
F A ROGUEOR SCANDAL-MONGER. HE IS DISHONEST.
1120 DATAA FALL IN FORTUNE.   RUIN OF SOMETHING ESTABLISHED.,SURPRISE AND WONDER
  LEADING TO DISQUIET.,THE END OF TROUBLES. ADVERSITY IS OVER.,PROSPERITY AND IN
-   CREASE. AN APPRECIATION OF BEAUTY.
1130 DATACONTRADICTION AND THETENDENCY OF OTHERS TO TRICK YOU.,APPREHENSION AND
THE FEAR OF DEFEAT.,PERPLEXITY. EMBARRASS-MENTS AND WORRIES.,QUARRELS ARISING FR
OMJEALOUSY OR ENVY.
1140 DATAOBSTACLES TO YOUR    IDEAS. POSSIBLE CALAMITY.,DIFFICULTIES AND THE INT
REGUE OF OTHERS AGAINST YOU., BAD NEWS OR ANNOUNCE-MENTS. INDECISION.,DISCORD. I
NTERRUPTIONOF THE PRESENT STATUS QUO.
1150 DATAPROBLEMS OF JEALOUSY - DECEIPT OR INFIDELITY.,"THE INFLUENCE OF A   GOO
D BUT SEVERE MAN. HE IS AUSTERE, YET TOLERANT."
```

PET Bloobers

by Richard Warner

Here's a fascinating — hypnotic, even — game for the PET. It includes instructions and details on converting it to run on old ROM machines.

```
10 REM
11 REM N.B. ALTER LINE 110 TO CHANGE
12 REM SCREEN SIZE.
13 REM H IS SCREEN WIDTH - MIN=3,MAX=38
14 REM V IS SCREEN DEPTH - MIN=3,MAX=23
15 REM BAD SUBSCRIPT ERROR MAY OCCUR
16 REM ON OLD ROM MACHINES IF H*V>256.
100 P=C:T=R:S=RND(-TI)
110 H=20:V=12
120 TP=32768+INT(19-H/2)+40*INT(12-V/2)
130 P2=TP+H+1
140 P3=TP+V*40+40
150 P4=TP+41+H+V*40
160 IN=166
161 REM OLD ROM'S IN=515
162 REM NEW ROM'S IN=166
170 SS=H*V-1
171 REM OLD ROM'S ONLY:IFSS>255THENSS=255
190 GOSUB1000
200 DIMC(3),D(3,1)
210 FORF=0TO3:READC(F),D(F,0),D(F,1)
230 NEXT
250 DATA85,1,40
260 DATA73,40,-1
270 DATA75,-1,-40
280 DATA74,-40,1
290 DIMP(SS),R%(SS),C%(SS),T%(SS)
300 PRINT"]"
310 FORF=TPTOP2:POKEF,102:NEXT
320 FORF=TPTOP3STEP40:POKEF,102:NEXT
330 FORF=P2TOP4STEP40:POKEF,102:NEXT
340 FORF=P3TOP4:POKEF,102:NEXT
400 C=INT(RND(1)*4)
410 R=INT(RND(1)+.5)
420 P=TP+INT(RND(1)*(H-2))+40*INT(RND(1)*(V-2))+82
430 IFPEEK(P)<>32THEN420
440 IFPEEK(P+D(C,0))<>32ORPEEK(P+D(C,1))<>32THEN420
450 S=0:T=0
500 POKEP,C(C):P(S)=P:T%(S)=T:R%(S)=R:C%(S)=C
510 S=S+1:P=P+D(C,R)
520 IFPEEK(P)<>32THEN600
530 T=INT(RND(1)+1.5)
540 IFR=0THEN570
550 C=C-T:IFC<0THENC=C+4
560 GOTO580
570 C=C+T:IFC>3THENC=C-4
580 IFT=2THENR=1-R
590 GOTO500
600 IFP=P(0)THEN700
610 GETZ$:IFZ$=" "THEN800
620 S=S-1:P=P(S)
630 IFT%(S)=0THEN690
640 R=1-R%(S):IFR=1THEN670
650 C=C%(S)-1:IFC<0THENC=C+4
660 GOTO680
670 C=C%(S)+1:IFC>3THENC=C-4
680 T=0:IFPEEK(P+D(C,R))=32THEN500
690 POKEP,32:GOTO610
700 T=C%(0)-C:IFT=0THEN610
710 IFR=0AND(T=-1ORT=3)THEN610
720 IFR=1AND(T=-3ORT=1)THEN610
750 T=TI:P=0
760 GETZ$:IFZ$>""THEN800
770 IFTI-T<600THEN760
800 S=S-1:POKEP(S),32
810 FORF=0TO50:NEXT
820 IFS=0THEN400
830 IFPEEK(IN)=6ORF=0THEN800
840 P=P(S):R=R%(S):C=C%(S):T=1:GOTO500
1000 PRINT"]";TAB(15);"BLOOBERS"
1010 PRINT"  BLOOBERS RANDOMLY GENERATES PATTERNS"
1020 PRINT"ON THE SCREEN.
1030 PRINT"  IF YOU GET BORED, PRESS THE SPACE BAR"
1040 PRINT"AND THE PATTERN WILL BACKTRACK UNTIL"
1050 PRINT"YOU TAKE YOUR FINGER OFF."
1060 PRINT"           PRESS ANY KEY TO GO"
1070 GETZ$:IFZ$=""THEN1070
1080 RETURN
```

PEEK and POKE for Apple Pascal

by Stephen Withers

Apple Pascal provides routines to control all the standard Apple hardware but controlling additional devices can be a problem. For example, the communications card initialises to 300 baud but should you want to use a 110-baud Teletype then you'll have to load the ACIA's control register with the appropriate value. Instead of writing a routine to do just this, why not make it as generalised as possible by adding a POKE routine? It's then only logical to complement this with a PEEK facility. Just a couple of notes to accompany this listing: using macros in such a short listing is obviously unnecessary but the Pascal Editor allows such items to be copied into a program from a library file; if the value to be POKEd is higher than 255 then the high order byte is ignored; the printer used to produce this listing reproduces the character '#' as '£'.

```
.MACRO  POP ;SAVE TWO BYTES FROM STACK. THE
                PARAMETER MUST BE A FREE PAGE 0
        ADDRESS. USEFUL FOR SAVING A SUBROUTINE'S
PLA             RETURN ADDRESS.
STA     %1
PLA
STA     %1+1
.ENDM
;
.MACRO  PUSH     ;THE COMPLEMENT OF POP
LDA     %1+1
PHA
LDA     %1
PHA
.ENDM
;
.MACRO  DISCAR   ;DISCARD 4 BYTE STACK BIAS
PLA
PLA
PLA
PLA
.ENDM
;
;
;ALLOCATE SOME STORAGE SPACE IN PAGE 0
RETURN    .EQU    0
ADDR      .EQU    2
;
;
.FUNC PEEK,1
;------------------------------
;EQUIVALENT TO BASIC'S PEEK, AND
; FUNCTION PEEK(ADDRESS:INTEGER):INTEGER;
;------------------------------
POP      RETURN   ;SAVE RETURN ADDRESS
DISCAR            ;DISCARD STACK BIAS
POP      ADDR     ;SAVE ADDRESS PARAMETER
LDA      £0       ;MSB OF RESULT
PHA               ; IS ALWAYS ZERO
TAY     ;ZERO Y FOR DUMMY INDEXED ADDRESSING
LDA     @ADDR,Y;GET CONTENTS OFLOCATION REQUESTED
PHA               ;SAVE LSB OF RESULT
PUSH     RETURN   ;PUT ADDRESS BACK ON STACK
RTS               ;ALL DONE!
;
;
.PROC POKE,2
;---------------------------------------
;EQUIVALENT TO BASIC'S POKE AND
; PROCEDURE POKE(ADDRESS,VALUE:INTEGER);
;---------------------------------------
POP      RETURN   ;SAVE RETURN ADDRESS
PLA               ;THAT'S THE BYTE WE WANT
TAX               ;SAVE IT
PLA               ;IGNORE HIGH BYTE OF VALUE
POP      ADDR     ;SAVE ADDRESS PARAMETER
LDA      £0;DUMMY INDEX FOR INDIRECT ADDRESSING
TAY
TXA          ;GET LOW BYTE OF VALUE PARAMETER
STA          @ADDR,Y ;DO THE POKE
PUSH     RETURN   ;RESTORE RETURN ADDRESS
RTS               ;AND FINISH...
.END
```

PET Demolition

by P L Brown

```
1 READA:IFA=0THEN3
2 GOTO1
3 FORL=832TO878:READX:POKEL,X:NEXTL
4 SYS(832):RESTORE
5 K=255
8 GOSUB3500
9 N=59464:POKE59467,16:POKE59466,1:POKEN,1:R=0
10 TI$="000000":T1=700:D=40:PRINT""
11 B1=33489:B2=33526
12 FORBB=1TO7
13 FORB=B1TOB2
14 POKEB,208:NEXTB:B1=B1+40:B2=B2+40:NEXTBB
30 T1=700-(P/2):A=INT((100)*RND(1)):FORPE=32848TO32888:
   IFPEEK(PE)=208THEN2000
31 NEXT:IFTI>T1THENGOSUB1000
32 POKE32887,31:POKE32847,32
40 IFA<51THENX=32809:M=1
50 IFA>50THENX=32846:M=-1
60 POKEX,42:POKEX-1,32:POKEX+1,32
70 GETA$
75 IFA$="E"THEN7000
80 IFA$=" "THENPOKEX,32:GOTO200
90 X=X+M
100 IFX>32846THENPOKEX,32:M=-1
110 IFX<32809THENPOKEX,32:M=1
120 GOTO60
200 X=X+D
210 IFPEEK(X)=208THENPOKEX-D,32:POKEX,32
    :POKE59464,K:P=P+1:GOTO500
220 IFX>33768THENPOKEX-D,32:POKEN,
    255:FORL=0TO500:NEXT:POKEN,1:GOTO30
230 POKEX,42:POKEX-D,32:GOTO200
500 READDA:IFDA=0THENRESTORE:K=255:POKEN,1:GOTO1200
510 IFPEEK(X-DA)=208THENP=P+1:POKEX-DA,32:POKEN,K:K=K-9:
    GOTO500
530 GOTO500
900 DATA-40,-1,1,39,41,42,38,83,82,78,77,124,123,
    117,116,165,164,156,155
901 DATA206,205,195,194,245,246,235,234,233,247,0
1000 R=R+1:IFR>7THEN1050
1010 PRINT:TI$="000000":RETURN
1050 IFR>13THENR=0
1100 PRINT:FORRR=33729TO33766
1110 POKERR,208:NEXT:TI$="000000":RETURN
1200 PRINT"POINTS:"P"          HIGH SCORE"HP"
     ":GOTO30
2000 FORB=32887TOPE+1STEP-1:      POKEB,31:NEXT
2001 FORB=0TO100:POKEPE,32
2010 POKEPE,224:POKE59464,B:NEXT
2020 POKE59464,1
2030 PRINT"";
2040 POKE525,0:PRINT"YOUR SCORE IS"P
2050 IFP=>HPTHENHP=P:PRINT"IT IS THE BEST SCORE
     SO FAR":GOTO2070
2060 PRINT"THE BEST SCORE IS"HP
2070 PRINT"        TO PLAY AGAIN PRESS ANY KEY       "
2075 PRINT"   TO END THE PGM PRESS  'E' "
2080 GETA$:IFA$=""THEN2080
2085 IFA$="E"THEN7000
2090 P=0:GOTO9
3500 POKE59409,52:PRINT"":FORA=0TO22
3510 PRINT"";
3520 NEXT:PRINT""
4000 PRINT""SPC(14);" DEMOLITION "
4010 PRINT""SPC(11);" BY  PETER WRIGHT "
4100 POKE59409,60:FORA=0TO3000:NEXT
5000 PRINT"               DO YOU NEED INSTRUCTIONS ?"
5010 GETA$:IFA$=""THEN5010
5020 IFA$="N"THENRETURN
5030 PRINT"THE OBJECT OF THE GAME IS TO DEMOLISH"
5040 PRINT"THE WALL AT THE BOTTOM OF THE SCREEN."
5050 PRINT"THIS IS DONE WITH THE '*'.
5060 PRINT"THE 'SPACE' KEY IS USED TO FIRE THE '*'"
5070 PRINT"AT THE WALL."
5080 PRINT"THE WALL WILL ADVANCE AND IF IT"
5090 PRINT"GETS AS HIGH AS THE ARROW ON THE RIGHT"
6000 PRINT"OF THE SCREEN IT IS THE END OF THE GAME."
6002 PRINT"THE STOP KEY HAS BEEN DISABLED AND"
6003 PRINT"THE 'E' KEY MUST BE USED IN ITS PLACE"
6010 PRINT"          PRESS ANY KEY TO PLAY
6020 GETA$:IFA$=""THEN6020
6030 RETURN
7000 POKE59467,0:POKE59466,0:POKE59464,0:SYS(845)
8001 DATA120,169,99,141,25,2,169,3,141,26,2,88,96,
     120,169,133,141,25,2,169,230
8002 DATA141,26,2,88,96,169,0,72,72,72,72,76,133,
     230,32,90,3,234,169,255,141,9
8003 DATA2,76,126,230
```

Superboard/UK101 Bug bypass

by Roger Derry

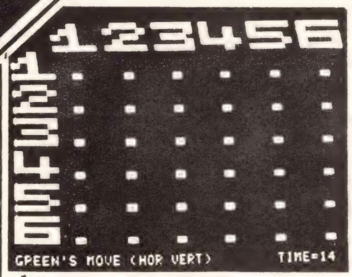
Officially, to associate a character string with a number, it is necessary to set up a string array and fill it via the usual READ and DATA statements. The disadvantages of this approach are twofold: first, the text appears twice in memory, and second, there is a bug in the garbage collection routine which causes these machines to hang up very easily when string arrays are used.

This program demonstrates a way of accessing the individual DATA statements without using string arrays. The secret lies in the fact that locations 143 and 144 contain the address of the next DATA statement. It becomes a simple matter to READ through the DATA list loading some or all of the DATA addresses into numeric variables. If these are subscripted then the addresses may be manipulated just like string variables but slower and using less memory. By POKEing 143 and 144 with the appropriate values DATA can be READ in any sequence, even backwards.

```
100 REM PROGRAMMABLE RESTORE DEMO.PROGRAM
110 REM ROGER DERRY
120 REM JULY 1980
125 :
126 :
127 REM INITIALIZE NUMERIC ARRAYS
128 :
130 DIM D(7,1),M(12,1)
136 :
137 REM D= DAYS OF WEEK  M= MONTHS OF YEAR
138 :
140 FOR X=1 TO 7
150 D(X,0)=PEEK(143):D(X,1)=PEEK(144)
160 READ A$
170 NEXT
175 :
180 FOR X=1 TO 12
190 M(X,0)=PEEK(143):M(X,1)=PEEK(144)
200 READ A$:NEXT
201 :
205 PRINT"ENTER 0 TO PRINT MONTHS BACKWARDS OR
210 INPUT "ENTER DAY (1-7) ";D
215 IF D=0 THEN 2000
220 IF D(1 OR D)7 THEN 210
222 :
225 INPUT "ENTER MONTH (1-12)";M
230 IF M(1 OR M)12 THEN 225
235 :
240 POKE 143,D(D,0):POKE 144,D(D,1)
250 READ D$
255 :
260 POKE 143,M(M,0):POKE 144,M(M,1)
270 READ M$
275 :
280 PRINT:PRINT:PRINTD$,M$
290 PRINT:PRINT:GOTO 210
300 :
301 :
1000 DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,
     THURDAY,FRIDAY,SATURDAY
1010 DATA JANUARY,FEBRUARY,MARCH,APRIL,MAY,JUNE
1020 DATAJULY,AUGUST,SEPTEMBER,OCTOBER,
     NOVEMBER,DECEMBER
1500 :
1501 :
2000 PRINT:PRINT:REM  PRINT MONTHS BACKWARDS!
2005 POKE 16,30:REM SET WIDTH FOR COMMA SPACING
2010 FOR X=12 TO 1 STEP -1
2020 POKE 143,M(X,0):POKE 144,M(X,1)
2030 READ M$:PRINTM$,
2040 NEXT
2050 PRINT:PRINT:PRINT
2060 GOTO 210
```

APPLE II

Brain Games-1, CS-4004 (16K)



1. Nuclear Reaction

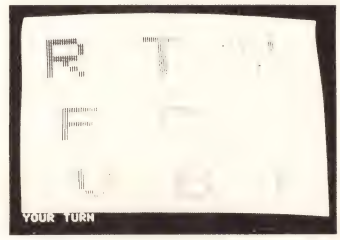
Nuclear Reaction is an exciting strategy game for two players. Each player, in turn, places a particle of radioactive material on a 6x6 board. When the number of particles at a location reaches its critical mass, it explodes, sending a particle to adjacent squares. As the board fills up, a single explosion can cause long chain reactions. The object of the game is to cause the right chain reaction to wipe out all your opponent's pieces. Nuclear Reaction is a game of skill, fast decisions, and quick reversals, making it fun to play for many hours. Action sound effects.

2. Dodgem

In Dodgem, two sets of pieces move at right angles across a checker style board. The object is to move all your pieces across the board and off the opposite edge. One player moves from the bottom to the top while the other moves left to right. You may play Dodgem against the APPLE or a friend. Six board sizes and action sound effects.

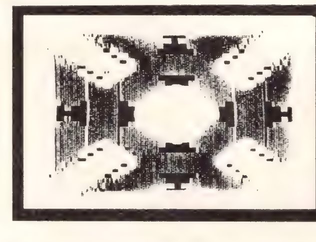
3. Dueling Digits

Do you have a good memory for sequences of numbers? Play Dueling Digits and find out. In this game, the computer presents a digit and tone, then erases it. You then type in the same digit. After each turn, the computer repeats all the previous digits and adds a new one, and you try to type in the entire sequence. The round is over when you make four mistakes. For up to four players. Two skill levels.



4. Parrot

Parrot is similar to Dueling Digits, but you try to remember sequences of letters and tones instead. Two skill levels.



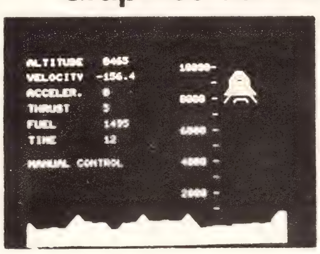
5. Midpoints and Lines

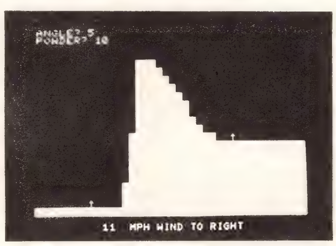
These two colorful graphics demonstrations will run continuously. Great for store displays, parties, and showing off your computer.

6. Tones

This program allows you to make your own music and sound effects with the game paddles. One paddle controls the pitch of the tone, the other controls the duration. No special hardware is required.

Sorcerer
Graphics Games-2, CS-5001 (8K)





1. LEM

In this graphic version of the popular real time lunar landing game you must land on the moon's surface with the lowest possible velocity. You can control the thrust of your retro-rockets with the number keys but you have a limited amount of fuel. The automatic pilot option can be activated and deactivated with the keyboard. You take a walk on the moon and plant a flag if you land successfully.

2. Pie Lob

This is a game in which two players lob pies at each other across a computer-generated hill. You choose the angle and the strength of the throw and then watch the trajectory of the lob across the screen. The computer changes the terrain and the wind speed in each game. Pie Lob makes good use of SORCERER's graphics.

3. Nuclear Reaction

Nuclear Reaction is an exciting strategv games for two players. Each player, in turn, places a particle of radioactive material on a 6x6 board. When the number of particles at a location reaches its critical mass, it explodes, sending particles to adjacent squares. As the board fills up, a single explosion can cause long chain reactions. The object of the game is to cause the right chain reactions to wipe out all your opponent's pieces. Nuclear Reaction is a game of skill, fast decisions, and quick reversals, making it fun to play for many hours.

4. Bounce

Bounce is an intriguing graphics demonstration which traces the path of a ball as it bounces around the screen.

5. Checkers

The SORCERER matches its strategy against yours in this popular game. The computer does not look more than one move ahead, hence the game is best suited for beginning players.

6. Dodgem

Dodgem is played on a checker-type board against the computer or another player. The object of the game is to block your opponent to slow him down. One player moves pieces from the bottom of the board to the top, and the other player moves from left to right, trying to get all the pieces off the board. This is a challenging game of strategy.

CHALLENGER

Graphics Games-3, CS-6001 (8K)

1. Tank Attack

Maneuver your tank around trees, houses and airplanes to destroy enemy guns. The guns fire back, and sometimes you get the distinct impression you have been lulled into a sense of security before getting blown up. A challenging real time game written by one of our most vicious programmers.

2. Dodgem

Dodgem is played on a checker-type board against the computer or another player. The object of the game is to block your opponent to slow him down. One player moves pieces from the bottom of the board to the top, and the other player moves from left to right, trying to get all the pieces off the board. A challenging strategy game.

3. Free For All

This game started as a joke, but it's for real! Two people compete against each other and the computer. A submarine, a destroyer, and a plane, criss cross the screen. Each may fire at the others. Free For All makes extensive uses of the Challenger's graphics.

4. Hidden Maze

In this game, two players (you, and the computer or a friend) compete to be the first to reach the opposite side of the maze. The maze is hidden at first and is only revealed as you run into parts of it. Swinging gates add an additional challenge to this game of memory and skill

Pursuit Games, CS-3004 (16K)

• Stock Car Race

Stock Car Race is a real time racing game on a road race circuit. Your high speed racer is controlled by the "arrow" keys, as you shift up and down through four gears. Take the turns slowly, "floor it" on the straights, but don't blow your engine!

• Maze

Maze for the Level II 16K machine is a high speed pursuit game. You are timed throughout your run and rated on the basis of elapsed time and the number of moves required to escape. A different maze every time. Nine skill levels.

• Indy Racer

Indy Racer is a real time racing game for the TRS-80. You're in the driver's seat of a red-hot Indy car, changing gears and weaving around the track as you pass your competitors. Indy Racer is similar to the popular arcade-style driving games.

• Depth Charge

As commander of a destroyer, your mission is to destroy as many enemy subs as possible. Move your ship back and forth on the water, positioning yourself over enemy subs as they cruise into range. Depth charges sink slowly, so timing and position are important in this re-creation of the Battle of the Atlantic.

• Kaleidoscope

This graphics demonstration program turns your TRS-80 into a computer age kaleidoscope. You enter the number of lines and size of the display to produce changing patterns on the video monitor. Truly hypnotizing, Kaleidoscope runs continuously to brighten up your home or office.

WRITE FOR...
FREE
SOFTWARE CATALOG

Strategy Games, CS-3005 (16K)

• Tunnel Vision

You are transported into a massive labyrinth and must find the exit or be lost forever. This is an excellent example of three dimensional perspective using TRS-80 graphics.

• Evasion

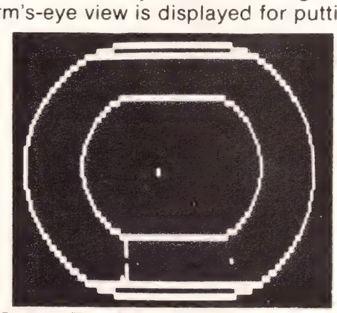
In this real time game, you are pursued around the game board by an evil-looking snake. Variations of play include two different speeds and hyper-jumps which randomly relocate you on the board. Looking for an escape? Try Evasion.

• Jigsaw

Jigsaw is a computer-age puzzle game making extensive use of TRS-80 graphics. The computer generates a random puzzle and puzzle board. Using a combination of deductive reasoning and luck you must fit the graphically represented puzzle piece into place.

• The Masters

Are you a wandering pro or just a Sunday golfer who would like to keep in practice? Once you're on the green, a worm's-eye view is displayed for putting.



• Motor Racing

Motor Racing combines real time racing action with advanced graphics functions. The graphics and animation make Motor Racing fun to watch as well as play.

Space Games-3, CS-3002 (16K)

• Ultra-Trek

Ultra-Trek is a fast-paced version of Star Trek, complete with "real time" action graphics, lasers, Nilon space mines, high energy photon torpedoes, enemy ships that move, and an experimental ray which does something different each time you use it. You must act quickly to save yourself and the Federation.

• Star Lanes

Imagine yourself the president of an intergalactic shipping company. If you're successful, you may be named Imperial Advisor on Economic Affairs. Entrepreneurs: to your ships.

• Star Wars

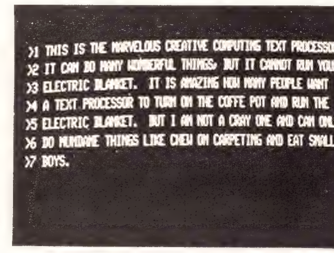
If you hate Darth Vader, you'll love Star Wars. This real time game is fun for aliens of all ages. May the Force be with you!

• Romulan

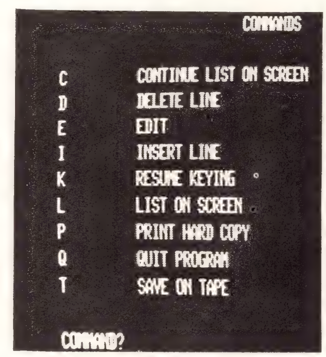
Your mission is to destroy an invading Romulan space craft. Maneuver through space and around stars looking for the deadly enemy, but be careful! The nasty Romulans fire back.

Text Processing, CS-3302 (16K)

This program turns a 16K, TRS-80 and lineprinter into a line oriented text-processing system.



Developed exclusively for the TRS-80, this program lets you use the computer to enter general text or business letters, edit and modify your work, save text on cassette tapes, and print out a perfect report, document, or letter every time.



Editing commands are similar to those used in Level II BASIC, so there are no complicated new commands to learn. Lines may be either inserted or deleted. A special format is available to speed entry of business letters. Final printout can be done in numbered pages and you may print multiple copies.

Phd Software
P.O. Box 1007, Brunswick, Victoria.

Please forward me the following Sensational Software.

Quantity	Item	Price
	CS 3002	\$14.95 + 50c p&p
	CS 3004	\$14.95 + 50c p&p
	CS 3005	\$14.95 + 50c p&p
	CS 3302	\$14.95 + 50c p&p
	CS 4004	\$14.95 + 50c p&p
	CS 5001	\$14.95 + 50c p&p
	CS 6001	\$14.95 + 50c p&p

I enclose cheque for $

Name..

Address..

....

.... Postcode..................

We warrant to the user of Sensational Software that it is free from defective materials and workmanship. If the product proves faulty, it will be replaced if returned to the address above.

String Function

by J. Di Stefano

String Manipulation Routines of the type described below are generally not provided in high level languages. It is very common in user oriented programs such as in CAI or gaming programs that user input be matched to some expected reply. If the reply is of an alphanumeric form the blanks in the input string are often not significant and must be removed before comparing the two strings. Similarly, lower case input 'yes' would not match the string 'YES' and thus the need to convert from lower case to upper case before the comparison is made.

The routines below provide for:

1. Remove all blanks from an argument string.
2. Remove leading blanks from an argument string.
3. Remove trailing blanks from an argument string.
4. Reduce multiple adjacent blanks to a single blank.
5. Convert lower case to upper case.
6. Remove carriage return, line feed, tab, and escape characters.

Before jumping to the subroutine two variables must be set up.

1. The Argument String (ARG$)
2. The Function Code (ARG)

The Function Code (ARG) may take any value less than 64 and greater than zero (0 ARG 64). For example, a Function Code of 4 (ARG @ 4) will remove trailing blanks from the string ARG$. But a Function Code of 6 (2&4) will remove both leading and trailing blanks. A Function Code of 22 (2&4&16) will remove leading and trailing blanks and convert any lower case text to upper case. The subroutine returns the new string ARG$ appropriately modified.

In the program below the subroutine is called at line 320 GOSUB 20010. The return string and its length is output by line 330. The sample run tests the various string functions possible. The code presented should run without modification on most Microsoft type systems.

```
80      REM     STRING.BAS
82      REM     ----------
84      REM
86      REM
100     REM     TEST PROGRAM
120     REM
200     T$="THIS  IS THE  TEST string" + CHR$(13)+ CHR$(10)+"
305 INPUT"ARGUMENT ";ARG
310     ARG$=T$
320     GOSUB 20010
330 PRINT"ARG =";ARG;".......";ARG$,LEN(ARG$)
350     GOTO 305
400     REM ****************************************
            *******************
410     REM ****************************************
            *******************
420     REM ****************************************
            *******************
19999   REM     SUBROUTINE MUST BE CALLED WITH  ARG  &  ARG$
                SUBROUTINE RETURNS  ARG$
20000   REM     ARG$    - ARGUMENT STRING
20001   REM     LARG    - LENGTH OF ARGUMENT
20002   REM     ARG     -FUNCTION CODE(MUST BE: 1,2,4,8,16,32
        OR ANY SUM OF THESE)
20003   REM
20004   REM     ARG  FUNCTION
                ---  --------
                 1   REMOVE ALL BLANKS
                 2   REMOVE LEADING BLANKS
                 4   REMOVE TRAILING BLANKS
                 8   REDUCE ANY MULTIPLE ADJACENT BLANKS TO
                     A SINGLE BLANK
                16   REDUCE LOWER CASE TO UPPER CASE
                32   REMOVE CR, LF, TAB, ESC
20010   IF ARG >63 THEN ARG=63
20012   DIV = 32
20015   IF ARG < DIV THEN DIV=DIV/2 :  GOTO 20015
20017   ARG=ARG-DIV : TARG=DIV : GOSUB 20050
20020   IF ARG > 0 THEN 20015
20030   RETURN
20050   BFLAG=0
20060   LARG=LEN(ARG$)
20065   TARG$=""
20070   T1ARG$=""
20080   BLANK$=" "
20100   IF TARG =  1 THEN 20200
20102   IF TARG =  2 THEN 20400
20104   IF TARG =  4 THEN 20600
20106   IF TARG =  8 THEN 20800
20108   IF TARG = 16 THEN 21000
20110   IF TARG = 32 THEN 21200
20199 REM  REMOVE ALL BLANKS
20200   FOR G=1 TO LARG
20210   T1ARG$=MID$(ARG$,G,1)
20220   IF T1ARG$ <> BLANK$ THEN TARG$=TARG$+T1ARG$
20230   NEXT G
20240   ARG$=TARG$
20250   RETURN
20398 REM
20399 REM  REMOVE LEADING BLANKS
20400   FOR G=1 TO LARG
20410   IF MID$(ARG$,G,1) <> BLANK$
        THEN ARG$=RIGHT$(ARG$,G) : RETURN
20420   NEXT G
20430   ARG$=CHR$(0)
20450   RETURN
20598 REM
20599 REM REMOVE TRAILING BLANKS

20600   FOR G=LARG TO 1 STEP -1
20610   IF MID$(ARG$,G,1) <> BLANK$ THEN ARG$=LEFT$(ARG$,G)
         : RETURN
20620   NEXT G
20630   ARG$=CHR$(0)
20650   RETURN
20798 REM
20799 REM REDUCE ANY ADJACENT MULTIPLE BLANKS TO A SINGLE
      BLANK
20800   FOR G=1 TO LARG
20810   T1ARG$=MID$(ARG$,G,1)
20820   IF T1ARG$ <> BLANK$ THEN BFLAG=0 :
        TARG$ = TARG$+T1ARG$: GOTO 20850
20830   IF T1ARG$ = BLANK$ AND BFLAG=0 THEN BFLAG=1:
        TARG$=TARG$+T1ARG$
20850   NEXT G
20860   ARG$=TARG$
20870   RETURN
20998 REM
20999 REM  REDUCE LOWER CASE TO UPPER CASE
21000   FOR G=1 TO LARG
21010   T1ARG$=MID$(ARG$,G,1)
21020   T=ASCII(T1ARG$)
21030   IF T > 96 AND T<123 THEN T1ARG$=CHR$(T-32)
21040   TARG$=TARG$+T1ARG$
21050   NEXT G
21060   ARG$=TARG$
21070   RETURN
21098 REM
21099 REM  REMOVE CR LF TAB ESC (ALLCHARS ASCII VAL<32)

21200   FOR G=1 TO LARG
21210   T1ARG$=MID$(ARG$,G,1)
21220   IF ASCII(T1ARG$) >31 THEN TARG$=TARG$+T1ARG$
21230   NEXT G
21240   ARG$=TARG$
21250   RETURN
32767    END
```

Several Sorts

by S. Zadarnowski

Five different methods of sorting are incorporated (and clearly labelled) in the listing. Use it as it stands or as a library of sorting routines for your own programs.

```
5 REM--* PRESENTED BY S. ZADARNOWSKI *
7 REM
10 REM--* FIVE METHODS OF SORTING *
20 REM
30 INPUT" SELECT METHOD # (1 TO 5)";S
40 GOSUB 600
50 ON S GOSUB 100,200,300,400,500
60 GOTO 700
90 REM
100 REM--* METHOD ONE..PARTNER SORT *
110 REM
120 Z=0
130 FOR M=1 TO N-1
140 IF A(M)<=A(M+1) THEN Z=Z+1 :GOTO 160
150 T=A(M):A(M)=A(M+1):A(M+1)=T
160 IF Z=N-1 THEN RETURN
170 NEXT M :GOTO 120
195 REM
200 REM--* METHOD TWO..DIMINISHING LOOP,
    VERSION 1 *
210 REM
220 FOR M=1 TO N
230 FOR L=1 TO N-M
240 IF A(L)<A(L+1) THEN 260
250 T=A(L):A(L)=A(L+1):A(L+1)=T
260 NEXT L: NEXT M
270 RETURN
290 REM
300 REM--* METHOD THREE..DIMINISHING LOOP,
    VERSION 2 *
310 REM
320 J=2
330 FOR M=1 TO N-1
340 FOR L=J TO N
```

PROGRAMS

```
350 IF A(M)>A(L) THEN T=A(L):A(L)=A(M):A(M)=T
360 NEXT L:J=J+1: NEXT M
370 RETURN
390 REM
400 REM--* METHOD FOUR..REDUCING
    INTERVAL SORT *
410 REM
420 M=N
430 M=INT(M/2)
435 K=N-M: J=1
440 IF M=0 THEN RETURN
445 I=J
450 L=I+M
455 IF A(I)<=A(L) THEN 475
460 T=A(I):A(I)=A(L):A(L)=T
465 I=I-M
470 IF I>=1 THEN 450
475 J=J+1
480 IF J>K THEN 430
485 GOTO 445
490 REM
500 REM--* METHOD FIVE..SORT AS YOU GO *
510 REM
520 INPUT" HOW MANY ELEMENTS ";N
530 DIM A(N)
540 FOR M=1 TO N
550 PRINT"ELEMENT #";M;
560 INPUT A(M): L=M
565 IF A(L)>A(L-1) OR L=1 THEN 580
570 T=A(L):A(L)=A(L-1):A(L-1)=T
575 L=L-1: GOTO 565
580 NEXT M: RETURN
590 REM
600 REM--* GET DATA TO SORT *
610 REM
620 IF S=5 THEN RETURN
625 INPUT" (Y)OUR DATA, OR (R)ANDOM DATA";R$
630 INPUT" HOW MANY ELEMENTS ";N
635 DIM A(N)
640 IF R$="Y" THEN 670
650 FOR M=1 TO N
655 A(M)=INT(RND(0)*100)
660 NEXT M: RETURN
670 FOR M=1 TO N
685 PRINT"ELEMENT #";M;
690 INPUT A(M)
695 NEXT M: RETURN
700 REM
710 REM--* PRINT SORTED LIST *
720 REM
730 FOR M=1 TO N
740 PRINTM;"..";A(M),
750 NEXT M
760 END
READY.
```

USER GROUPS INDEX

VICTORIA

MICOM-80
A Special Interest Group of Melbourne's TRS-80 and System 80 Users Group can be contacted on 560-8132 or write to the Group Chairman, Len Saunders, at 9 Currajong Avenue, Glen Waverley, 3150. The group meets on the third Saturday of each month at the Ausom Users Hall.

Geelong Computer Club
Interested people should write to The Geelong Computer Club, P.O. Box 6, Geelong, 3220.

O.S.I./6502 Users Group
For information concerning this newly formed group, please contact Ian Eyles, 10 Forbes St., Essendon, 3040. Phone 375-3478 A.H.

Peninsula Group
This group meets at State College Frankston, on the second Tuesday of each month except during January. Those interested may contact M.G. Thompson on 772-2674.

AUSOM
Apple Users' Society of Melbourne. Contact Mr. David Turk of Computerland Melbourne.

S.M.U.G.
To find out more about this group of SORD M100 users, contact Mr. Robin Miller, 60 Winmalee Drive, Glen Waverley, 3150.

Compucolor Users' Group
Write to Mr. L. Ferguson of 12 Morphett Avenue, Ascot Vale for all the information necessary.

S.C.U.A.
Sorcerer Computer Users (Australia). Further details may be obtained from the Secretary, S.C.U.A., P.O. Box 144, Doncaster, 3108.

Commodore Computer Users Association
Nicki Saunders on 614-1433 during business hours, has all the information.

National Sinclair ZX80 Users Club
Tips and discussions of ZX80, sample programs, programming tips, discussions and news of developments in the U.K. and U.S., and a market place for goods and facilities specially provided for the ZX80. Write for free introductory newsletter: 24 Peel Street, Collingwood, 3066.

NEW SOUTH WALES

Australasia ZX80 Users' Group
This recently formed group aims to promote the use and growth of the Sinclair ZX80. A newsletter with listings of original programs submitted by members and friends, handy subroutines and other programming aides, as well as reviews on purchased software — hardware will be produced.

Anyone interested in the ZX80 may contact Tony Mowbray, 87 Murphys Ave., Keiraville, 2500. Phone (042) 28-5296.

Commodore User Group
For more details, drop a line to Mr. John Guidice, C/- The Commodore Users Group, G.P.O. Box 4721, Sydney, 2001.

Compucolor Users' Group
If you are interested, Andrew MacIntosh of 91 Regent Street, Chippendale, is the man to see.

80AT
Mail enquiries to 80AT, C/- Planet 3 Systems, 47 Birch St., Bankstown, 2200.

New South Wales Apple Users' Group
For more information write to J. Smith, 4/15 Raine St., Wollahra, 2025.

QUEENSLAND

Brisbane Youth Computer Group
Mr. A. Harrison, P.O. Box 396, Sunnybank, 4109, should be contacted for more information.

IREE Microcomputer Interest Group
No meetings will be held in January. Details on club membership etc. may be obtained from The Secretary, N. Wilson, P.O. Box 81, Albion, 4010. Phone 356-6176.

Ohio Superboard User Group
For membership and newsletter information, please send a large S.A.E. to Ed Richardson, 146 York St., Nundah, 4012.

SOUTH AUSTRALIA

TRS-80 Users' Group
To obtain details contact Mr. G. Stevenson of 36 Sturt Street, Adelaide, 5000

Commodore Computer Users' Association of S.A.
This group meets at 7.30 p.m. on the first Tuesday of each month at the Adelaide University Union Building. Further information is available from Earle Rowan, Commodore Computer Users' Association of S.A., P.O. Box 60, Clarence Gardens, 5039.

A.C.T.

MICSIG
Further information concerning MICSIG, from the Registrar, MICSIG, C/- P.O. Box 446, Canberra City, 2601.

TASMANIA

TEMOS
All enquiries are welcome and may be directed to John Stephenson, President, 4 Melinga Place, Taroona, 7000. Ph: 27-8770.

WESTERN AUSTRALIA

Apple Users Club of W.A.
The acting secretary/treasurer, Tim Russell may be contacted C/- P.O. Box 38, Willetton 6155 or 332-1726 A.H.

NEW ZEALAND

Wellington Microcomputer Society Inc.
Write to Lindsay Williams, 2 Pope Street, Plimmerton, New Zealand.

ADVERTISERS INDEX

ASP Microcomputers 33
Anderson Digital Equipment 36
C.I.S.A. 40
Commodore 4
C.R.G. 8
Computerland . . . . . IFC 2,IBC
Computerware 50
Datasoft 53
Deforest Software . . . . . Insert
Edible Electronics 10
Electromedical Engineering 52
Informative Systems 18
Logic Shop 6
Looky Video 18
MS Microsoftware 60
McGills Authorized Newsagency 36
Melbourne House 24,25
Mensa Computer Systems 49
Microcomputer House 13
MicroPro Design 26
micro Educational 10
Peter Hatcher 38,39
Phd Software 56,57
Seahorse Software 50
Sinclair Equipment (Australasia) 14,16
System Software 20
TCG Ohio Scientific . . . . . OBC